ON THE ROBUSTNESS OF CLUSTERED SENSOR NETWORKS

A Dissertation

by

JUNG JIN CHO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2007

Major Subject: Industrial Engineering

ON THE ROBUSTNESS OF CLUSTERED SENSOR NETWORKS

A Dissertation

by

JUNG JIN CHO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Yu Ding
Committee Members,	Amarnath P. Banerjee
	Richard M. Feldman
	Illya V. Hicks
	Bani K. Mallick
Head of Department,	Brett A. Peters

December 2007

Major Subject: Industrial Engineering

ABSTRACT

On the Robustness of Clustered Sensor Networks. (December 2007)

Jung Jin Cho, B.S., Seoul National University;

M.S., Seoul National University

Chair of Advisory Committee: Dr. Yu Ding

Smart devices with multiple on-board sensors, networked through wired or wireless links, are distributed in physical systems and environments. Broad applications of such sensor networks include manufacturing quality control and wireless sensor systems. In the operation of sensor systems, robust methods for retrieving reliable information from sensor systems are crucial in the presence of potential sensor failures. Existence of sensor redundancy is one of the main drivers for the robustness or fault tolerance capability of a sensor system.

The redundancy degree of sensors plays two important roles pertaining to the robustness of a sensor network. First, the redundancy degree provides proper parameter values for robust estimator; second, we can calculate the fault tolerance capability of a sensor network from the redundancy degree. Given this importance of the redundancy degree, this dissertation presents efficient algorithms based on matroid theory to compute the redundancy degree of a clustered sensor network. In the efficient algorithms, a cluster pattern of a sensor network allows us to decompose a large sensor network into smaller sub-systems, from which the redundancy degree can be found more efficiently.

Finally, the robustness analysis as well as its algorithm procedure is illustrated using examples of a multi-station assembly process and calibration of wireless sensor networks.

To my parents

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Ding, for his guidance, support, and advice throughout the course of this research. I would also like to express appreciation to my committee, Dr. Banerjee, Dr. Feldman, Dr. Hicks, and Dr. Mallick, for their advice and support.

Many thanks also go to Advanced Metrology lab members and Young-Myoung; my time at Wisenbaker building was pleasant and enjoyable because of their friendship. I also want to thank Tai and Yo for making my time in Texas happy and Samantha for proofreading.

My deepest gratitude goes to my parents for their endless love, support, and encouragement, which I cannot appreciate enough, and to my beloved wife and precious daughter for their love.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Recent innovations in sensor technology make it possible to deploy large number of sensors to monitor multiple target signals. As the number of deployed sensors gets large, a massive amount of information, related to the monitored targets, becomes available in real-time. One critical issue in the reliable operation of a distributed sensor network is its capability of functioning properly in the presence of sensor failures. Given the sheer number of sensors in a large scale network and the harshness of the environment in which sensors are deployed, the chances are actually so high that some sensors malfunction during their service lives. Without isolating and eliminating sensor anomalies, malfunctioning sensors could mislead our understanding about the monitored targets.

The redundant information from networked sensors gives a chance to eliminate sensor anomalies. Suppose that redundant sensors are installed for monitoring a target. Then, by cross-referencing the measurements from different sensors regarding the same target, we may tell which sensor is working properly and which one is not. Apparently, redundant measurements are essential to achieving the robustness of a sensor network.

The author's research focuses on identifying the redundancy degree of sensor measurements and subsequently quantifying the robustness of a sensor network. In this dissertation, the author addresses two issues regarding the robustness of sensor networks: 1) redundancy and robustness measure of a sensor network; 2) a robust procedure, which is less sensitive to sensor anomalies, to estimate the status of mon-

---

The journal model is *IEEE Transactions on Automatic Control.*

itored targets. In this dissertation study, a particular emphasis is put on the sensor network that has a cluster pattern.

A. Sensor network modeling

Modeling of sensor networks requires knowledge of the relationship between sensor measurements and monitored target signals. In many applications of sensor networks, two modeling tools are broadly used: a graph-based network representation and a linear model such as the observation equation in a typical linear state space representation [1]. In this dissertation, the author focuses on the sensor network that can be adequately modeled using a linear model such that

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}, \tag{1.1}$$

where $\mathbf{y} = (y_1, \ldots, y_n)^T$ is an n-dimensional vector of measurements or observations, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is a p-dimensional vector of unknown parameters, $\mathbf{e} = (e_1, \ldots, e_n)^T$ is the random errors, and $\mathbf{X} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T)^T$ is the $n \times p$ design matrix. Typically, the error term $\mathbf{e}$ is assumed to be normally distributed with zero mean and covariance matrix $\sigma^2\mathbf{I}$.

In fact, many graph-represented sensor network can also be converted to Model (1.1) using incidence or adjacent matrices [2]. Figure 1 shows a graph representation of a sensor network for a process flow network. In Figure 1, the directed edges of the graph represent the streams in the flow network, and the nodes represent the tanks and junctions. In the flow network, the streams are measured by sensors and denoted by $\mathbf{y}$ in Model (1.1). The inventory changes of the tanks and junctions, in the flow network, are the monitored parameters and denoted by $\boldsymbol{\beta}$. The relationships between $\mathbf{y}$ and $\boldsymbol{\beta}$ that are represented in Figure 1 can also be expressed using Model (1.1)

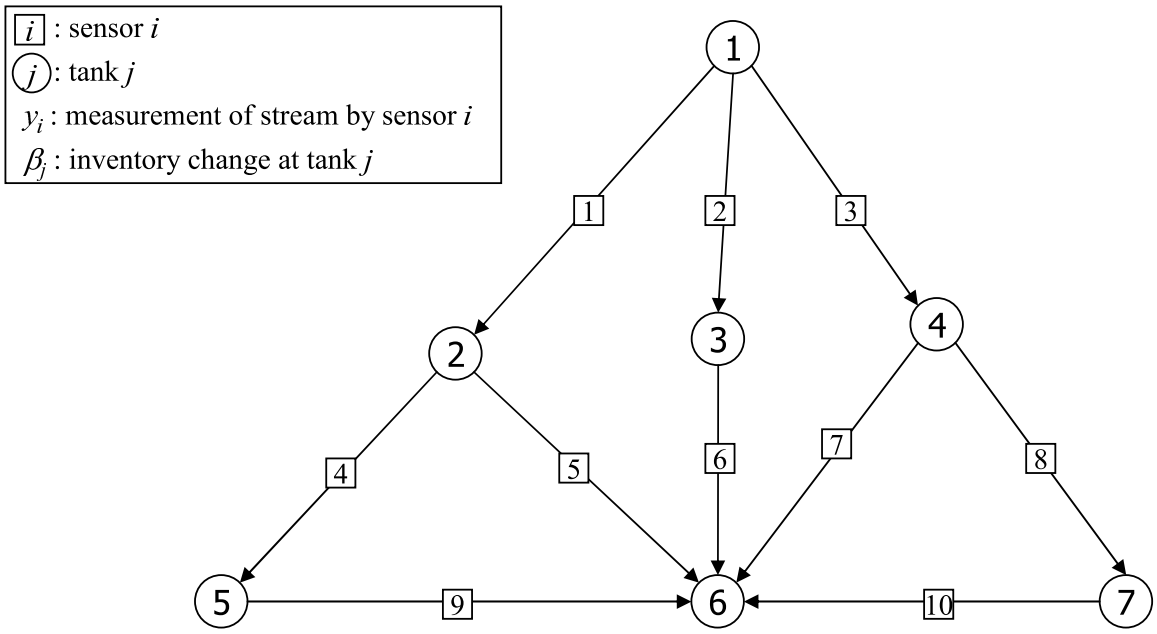


Fig. 1. Graph representation of a flow network

of which the design matrix is the incidence matrix of the directed graph in Figure 1 such that

$$\mathbf{X} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Obtaining the design matrix $\mathbf{X}$ in Model (1.1) is application-specific. This dissertation includes examples of Model (1.1) of a sensor network on a multi-station assembly process and wireless sensor networks in Chapter V. Many other sensor networks are also modeled using Model (1.1). Examples include, but are not limited to,

those in manufacturing processes [3, 4, 5], electrical power systems [6], power plant instrumentations [7], and navigation facilities [8].

B. Sensor redundancy degree

For the sensor networks represented by Model (1.1), prior research studied the degree of redundancy associated with sensor networks [9, 10, 11]. A common definition of the redundancy degree, denoted by $\eta(\mathbf{X})$, is as follows:

$$\eta(\mathbf{X}) = \min\{d - 1 : \text{ there exists } \mathbf{X}_{(-d)} \text{ s.t. } r(\mathbf{X}_{(-d)}) < p\}, \tag{1.2}$$

where $\mathbf{X}_{(-d)}$ is the reduced matrix after deleting d rows in $\mathbf{X}$, and $r(\mathbf{X})$ is the rank of $\mathbf{X}$. The issue on the redundancy degree $\eta(\mathbf{X})$ is that the redundancy degree is almost impossible to obtain for a large-size sensor network since no polynomial time algorithm is known. In order to assess the redundancy degree of a large-size sensor network, using matroid theory, the author devises algorithms that decompose a large-size sensor network into small subnetworks based on the cluster pattern. The details about such algorithms and related theorems are presented in Chapter III with brief introduction of matroid theory.

In the dissertation, the author proposes a robustness measure, the *fault tolerance capability*, of a sensor network and establish a functional relationship between the proposed fault tolerance capability and the redundancy degree. Intuitively, the fault tolerance capability is defined as the maximum number of sensor anomalies that the sensor network can tolerate before the whole system breaks down (rigorous mathematical definition is presented in Chapter IV). This dissertation shows that the fault tolerance capability cannot exceed the half of the redundancy degree $\eta(\mathbf{X})$ of a sensor network.

C. Related work

This dissertation consists of two parts. The first part identifies the redundancy degree efficiently using decomposition of a clustered sensor network, and the second part computes the fault tolerance capability and devises robust estimation of $\boldsymbol{\beta}$ using the computed redundancy degree. In this section of the dissertation, a comprehensive literature review related to the two topics is presented.

1. Redundancy and observability

In chemical engineering literature, the first paper about identifying observability and redundancy was published in 1981 [9]. The redundancy defined in [9] simply indicates the existence of redundant sensors; however, the concept of the redundancy degree was not introduced yet. Later, Krestsovalis and Mah [12, 13] proposed an algorithm to identify the redundancy of a sensor network, but their research was focused on the graph modeling; thus, the redundancy degree was still not introduced. The degree of redundancy was first introduced in [10]. Also, Ali and Narsmihan studied the redundancy degree and computed the reliability using the redundancy degree [14, 15].

In electrical engineering literature, the studies on the redundancy have been performed under the name of the fault tolerance study [11, 16]. However, those studies assume that sensor failure is 0–1 type and immediately identifiable; thus, the fault tolerance in those literatures is equivalent to the redundancy in chemical engineering literature.

2. Hardware-based approaches for detecting and eliminating sensor anomalies

In order to detect and eliminate sensor anomalies, hardware-based approaches have been traditionally employed, e.g., using off-line gauge repeatability and reproducibil-

ity (R&R) calibration [17] or built-in test equipment [18, 19, 20]. However, as a sensor network becomes very large, these hardware-based approaches become very costly and time consuming [21]. Therefore, the author's research excludes hardware-based approaches, but is focused on using redundancy in measurements from sensors.

3. Fault tolerance techniques for sensor networks

The fault tolerance capability of a sensor network is commonly known as the ability to sustain functionality without any interruption due to sensor failures [11, 22]. The fault tolerance capability has been quantified by the reliability [22, 23], the mean time to failure (MTTF) [11], or the number of possibly faulty sensors [24]. In [11, 22, 23], a sensor failure is assumed to be immediately identifiable, so the robustness study in those literatures do not address detecting and eliminating sensor failures. Marzullo [25] proposes the interval model to represent sensor networks and a fault-tolerant algorithm for identifying sensor anomalies. Later, several researchers also proposed fault-tolerant algorithms using the interval model, but the quantification of the fault tolerance was not mentioned yet [26, 27]. Jayasimha [24] treated the maximum number of sensor anomalies, which his fault-tolerant algorithm can identify using the interval model, as the fault tolerance capability. This stream of research starting from [25] does not consider multiple source signals.

4. Robust regression

It comes as no surprise that the proposed research is related to the robust regression since Model (1.1) is mathematically equivalent to those used in a linear regression analysis. If we treat sensor anomalies as outliers in statistical analysis, robust regression analysis that can handle the existence of outliers may be applied to a sensor network. The basic idea of robust regression is to limit the influence of outliers

[28, 29, 30, 31], or to utilize a subset of measurements that are more likely to be correct readings [32, 33, 34], or to use a combination of both [35, 36, 37, 38]. The robustness of a regression method is characterized by the breakdown point [39], which is the smallest fraction of outliers that can ruin an estimator. The higher the breakdown point value, the more robust the estimator is. In this dissertation, the author discusses the relationship between the degree of redundancy and the breakdown point value.

There have been a few papers (e.g., [40, 41]) on the robustness of a regression considering the inherent linear dependences in the row vectors of $\mathbf{X}$. In [40, 41], finding the linear dependences in the row vector of $\mathbf{X}$ is actually equivalent to computing the redundancy degree. Nevertheless, the majority of the studies on robust regression in statistics literature disregards the inherent linear dependences in the row vectors of $\mathbf{X}$. Mili and Coakley [41] demonstrated that robust estimators lose robustness if robust estimators blindly assume that $\eta(\mathbf{X}) = n - p$ as suggested in [32, 33, 34, 35, 36, 37, 38]. In [41], a quantity, namely L, is used to characterize the linear dependence relationship in $\mathbf{X}$, and the breakdown point associated with robust estimators is proved to be related with L. This dissertation shows that the quantity L is actually a simple function of the redundancy degree $\eta(\mathbf{X})$ and consequently, shows that finding the redundancy degree is essential to applying robust estimators and analyzing the robustness of the applied robust estimators.

5. Gross error detection

In sensor network applications, identifying the failed sensors is referred to as gross error detection [21, 42, 43, 44]. Gross error detection is basically to test, detect, and subsequently discard outliers in the measurements. The procedure generally starts with all the measurements and often runs recursively. If used carefully, robust

Table I. Summary of related work

Classifications	*Measures/Techniques*	*Publications*
Observability/Redundancy	Existence of Redundancy	[9] [12] [13] [16]
	Redundancy Degree	[10] [11] [14] [15]
Hardware-based approaches	Offline R&R	[17]
	Built-in Equipment	[18] [19] [20]
Fault Tolerance Techniques	Reliability	[22] [23]
	MTTF	[11]
	Number of faulty sensors	[24]
	Fault Tolerant Algorithm	[24] [25] [26] [27]
Robust Regression	Bounded Influence	[28] [29] [30] [31]
	Subset of Measurements	[32] [33] [34]
	Combinations	[35] [36] [37] [38]
	Breakdown point	[39] [40] [41]
Gross Error Detection	Identifying Sensor Anomaly	[21] [42] [43] [44]

regression and gross error detection oftentimes lead to very similar results (please see the example in Section 8D in [45]). In addition, the results from robust regression could help gross error detection by using a robust estimator as the starting point to detect outliers [6].

6. Summary of literature review

Table I summarizes the aforementioned literature. The differences between the research achievement in this dissertation and the research work summarized in Table I come from modeling of sensor networks and developing algorithms for the robustness

Table II. Comparison between related work and the research work in this dissertation

Publications	*Modeling*	*Algorithms for Robustness Measures*
[25] [26] [27]	Interval model	No robustness measure
[28] [29] [30] [31]	Matrix	No robustness measure
[17] [18] [19] [20]	Hardware-based	No robustness measure
[9] [10] [11] [12] [13] [14] [15] [16]	Graph/Matrix	Exhaustive search
[22] [23]	Graph	Efficient algorithm
[11]	Matrix	Exhaustive search
[24]	Interval model	Fault tolerant algorithm
[32] [39] [40] [41]	Matrix	No algorithm for breakdown point
This dissertation	Matrix/Matroid on clustered networks	Efficient Algorithm using decomposition

measures. Table II compares the research work in this dissertation with related work. This dissertation explains connections between the robustness measures in aforementioned different research communities (e.g., redundancy and breakdown point) and proposes the fault tolerance capability for sensor network applications.

D. Outline of the dissertation

This dissertation develops a framework that identifies a redundancy degree of sensor measurements and subsequently quantifies robustness of a sensor network. In Chapter II and III, this dissertation proposes techniques associated with computing the redundancy degree $\eta(\mathbf{X})$. Chapter IV deals with the robustness measures of a sensor

network, and Chapter V presents the demonstration of robustness analysis presented in previous chapters. The outline of this dissertation is illustrated in Figure 2. The organization of this dissertation is as follows.

Chapter I describes the motivation of the research and states the objective of the dissertation. A comprehensive review of the related work as well as the organization of this dissertation is also included in Chapter I.

Chapter II explains what a clustered sensor network is, and how Model (1.1) represents the cluster pattern in a sensor network. Chapter II presents two methods that identifies a cluster pattern in a sensor network.

Chapter III presents algorithms that obtain the redundancy degree $\eta(\mathbf{X})$ by decomposing a sensor network into smaller subsystems based on its cluster pattern. Chapter III also includes an introduction of matroid theory and supporting theorems.

Chapter IV introduces robustness measures, the breakdown point and the fault tolerance capability, and robust estimation procedures for achieving the maximum fault tolerance capability. Chapter IV shows that computing the redundancy degree $\eta(\mathbf{X})$ is essential for finding the robustness measures and devises robust estimation procedure.

Chapter V demonstrates the robustness analysis developed in the previous chapters using multi-station assembly processes and calibration problems for the localization of wireless sensor networks.

Finally, Chapter VI summarizes this dissertation with concluding remarks and provides recommendations for future work.

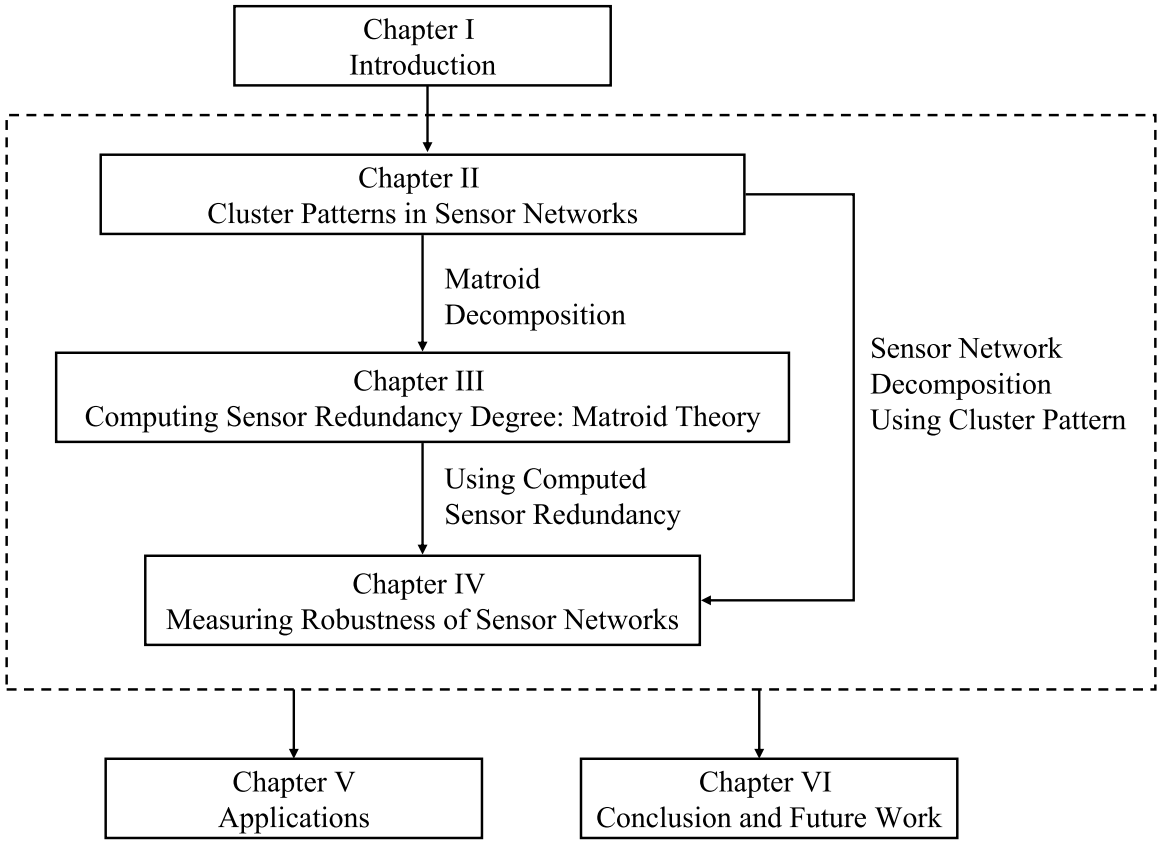


Fig. 2. Outline of the dissertation

CHAPTER II

CLUSTER PATTERNS IN SENSOR NETWORKS

The research work in this dissertation is focused on clustered sensor networks; cluster patterns in a sensor network allow us to decompose a large-size sensor network into smaller subsystems so that we can enhance the computation efficiency in obtaining the fault tolerance capability and devising robust estimation procedure. This dissertation refers to a *cluster*, as a set of sensor nodes in a sensor network, where relatively many links or communications exist within the set of sensor nodes; few links or communications exist between clusters.

We can easily find such cluster patterns in many sensor network applications. For example, wireless sensor networks embed cluster patterns due to the limited power supply on individual sensor nodes. Wireless sensor nodes, which are usually battery-powered, rarely communicate with all other sensors in a network. Instead, the whole network is usually grouped into clusters. A sensor node mainly communicates with other sensors of the same cluster. The between-cluster communications are limited to a few cluster-heads or nodes that are close to another cluster.

A cluster pattern in a sensor network usually causes the design matrix $\mathbf{X}$ to be a sparse matrix; a linear model with a sparse design matrix is called *structured linear model*. In [41], Mili and Coakley stated that structured linear models *"constitute a broad family of regression problems that are especially encountered in the engineering fields and physical sciences."*

The cluster pattern in a sensor network typically causes the design matrix $\mathbf{X}$ to have a certain pattern as well. Section II.A presents how $\mathbf{X}$ reflects the cluster pattern. After that, Section III.B and III.C present two methods that find a cluster pattern of a sensor network by analyzing $\mathbf{X}$.

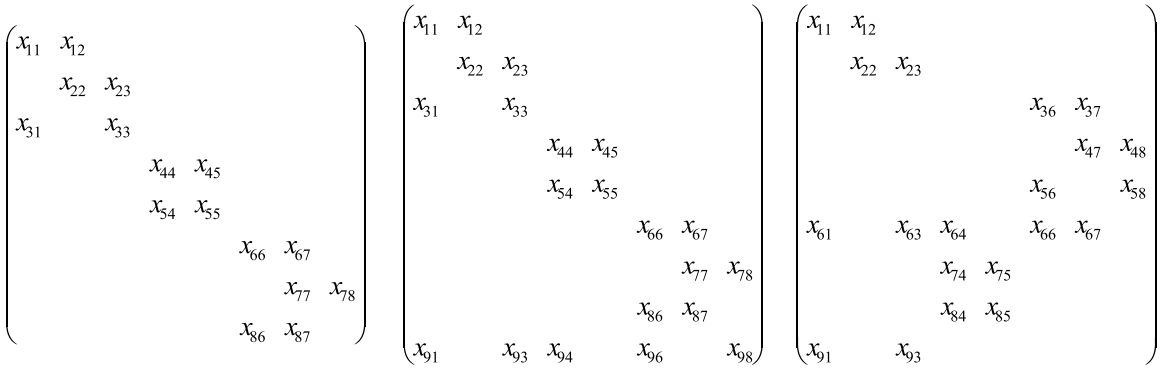


Fig. 3. The design matrix structures, where x_{ij} represents a nonzero element at ith row and jth column; elsewhere, and the elements are zeros.)

A. Bordered block form and clustered sensor network

Suppose a design matrix $\mathbf{X}$ is comprises a set of k disjoint submatrices, such that

$$\mathbf{X} = \begin{pmatrix} \mathbf{B}_1 & & \\ & \ddots & \\ & & \mathbf{B}_k \end{pmatrix}, \tag{2.1}$$

where $\mathbf{B}_1, \ldots, \mathbf{B}_k$ are nonzero matrices; then, X is said to be in the *block form*. Figure 3(a) shows an example of a matrix in the block form. For the design matrix $\mathbf{X}$ in the block form, it is easy to see that the redundancy degree $\eta(\mathbf{X})$ is simply the minimum among $\eta(\mathbf{B}_1), \ldots, \eta(\mathbf{B}_k)$ since the linear model (1.1) can be divided into k separate models, where the design matrices are $\mathbf{B}_1, \ldots, \mathbf{B}_k$.

In general, a design matrix in the block form is not very common; a more common manifestation of a sparse design matrix usually takes the format of the *bordered block*

form (BBF) such that

$$\mathbf{X} = \begin{pmatrix} \mathbf{B}_1 & & \\ & \ddots & \\ & & \mathbf{B}_k \\ \mathbf{S}_1 & \dots & \mathbf{S}_k \end{pmatrix}, \tag{2.2}$$

where $\mathbf{B}_1, \dots, \mathbf{B}_k, \mathbf{S}_1, \dots, \mathbf{S}_k$ are nonzero matrices. Figure 3(b) illustrates a bordered block form.

The bottom rows of a matrix in (2.2), $\mathbf{S}_1, \dots, \mathbf{S}_k$, illustrated as the bottom row in Figure 3(b), is called the *border row*. Of course, a sparse design matrix, which is in the bordered block form, could take far more complicated appearance such as the one in Figure 3(c). In a bordered block form, the common feature of a matrix is that once the border rows are identified and removed, the rest of the matrix can be decomposed into smaller size disjoint submatrices.

B. Hypergraph method for finding bordered block form

There are a few research papers reporting methods on permuting a matrix $\mathbf{X}$ into a border block form. Ferris and Horn [46] proposed a two phase approach to find a bordered block structure. Ayknant et al. [47] used hypergraph models to change the permutation problem to an k-way hypergraph partitioning problem. Tools such as hMeTis [48] and PaToH [49] provide very quick and stable results in solving the k-way hypergraph partitioning problems.

A hypergraph $H = (V, N)$ is defined as a set of vertices V and a multiset of hyperedges N, which are a subset of V. *Hypergraph representation* of $\mathbf{X}$ is a $H = (V, N)$ such that V is a set of column labels of $\mathbf{X}$ and $N_i \in N$ contains the vertices corresponding to the columns that have a non-zero entry in row i. Let $j \in V$. Then, $j \in N_i$ if and only if x_{ij} is non-zero, where x_{ij} is the (i, j)-entry of $\mathbf{X}$.

For example, suppose

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 0 & x_{14} \\ x_{21} & x_{22} & 0 & 0 \\ x_{31} & 0 & x_{33} & 0 \\ 0 & 0 & 0 & x_{44} \\ 0 & x_{52} & 0 & 0 \\ x_{61} & x_{62} & x_{63} & x_{64} \end{pmatrix},$$

where x_{ij} are nonzero elements. The hypergraph representation of $\mathbf{X}$ is $H = (V, N)$ such that $V = \{1, 2, 3, 4\}$ and $N = \{\{4\}, \{1, 2\}, \{1, 3\}, \{4\}, \{2\}, \{1, 2, 3, 4\}\}$. Figure 4 illustrates H. In Figure 4, the circled numbers (①, ②, ③, and ④) represent the vertices in V, and solid lines represent the hyperedges. For example, the rightmost hyperedge connecting ① and ③ represents $\{1, 3\}$, which corresponds to the third column in $\mathbf{X}$. Dashed ellipses show resulting partitions after removing hyperedge $\{1, 2, 3, 4\}$. The k-way hypergraph partitioning problem is to find a set of hyperedges of a minimum size whose removal disconnects the k vertex parts of the hypergraph. Using one of the aforementioned k-way hypergraph partitioning tools, we can see that the hypergraph H is disconnected by deleting the hyperedge $\{1, 2, 3, 4\}$, and the resulting partitions are $\{1, 2, 3\}$ and $\{4\}$. In Figure 4, these two partitions are contained by two dashed ellipses. In fact, the hyperedge $\{1, 2, 3, 4\}$ corresponds to the border rows of $\mathbf{X}$ and the partitions correspond to the blocks. Then, in the above example of $\mathbf{X}$, the separating set S is $\{6\}$, which corresponds to the hyperedge $\{1, 2, 3, 4\}$. Since we find two partitions, there are two blocks.

The aforementioned tools for solving the k-way hypergraph partitioning problem need users to specify the resulting number of blocks, k. The author recommends using $k = 2$ since it finds the smallest S, which usually benefits the algorithms that

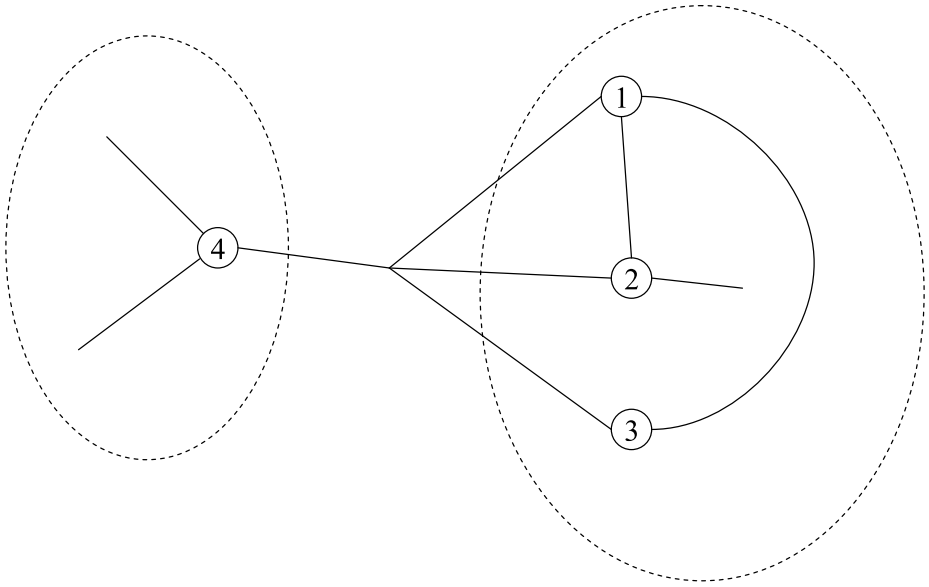


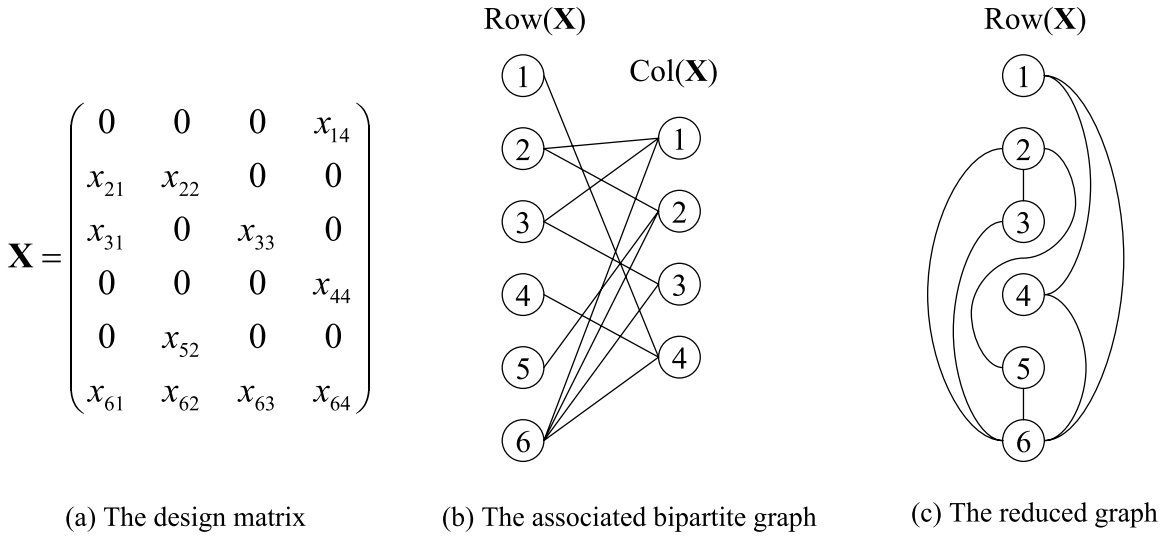
Fig. 4. Hypergraph representation

computes the redundancy degree $\eta(\mathbf{X})$ in Chapter III.

C. Reduced-graph method for finding bordered block form

In terms of permuting a sparse matrix into a bordered block form, several research groups in parallel computing have developed effective methods (e.g. [46, 47]), and a hypergraph-based method is presented in Section II.C. However, those approaches are quite involved and using them needs several inputs, such as the number of blocks and balancing criteria for regulating relative sizes of blocks, which may not sound intuitive to practitioners outside the area of parallel computing. By comparison, the reduced-graph approach is easier for practitioners to understand and implement. It automatically identifies the structure in a matrix and yields a bordered block form with the smallest border rows.

For the purpose of identifying a cluster pattern in a sensor network, we use a



(a) The design matrix (b) The associated bipartite graph (c) The reduced graph

Fig. 5. A bipartite graph representation of $\mathbf{X}$, where x_{ij} represents a nonzero element at ith row and jth column

bipartite graph representation of a matrix. A *bipartite graph*, also called a *bigraph*, is a graph whose vertices are partitioned into two disjoint sets, represented by V^+ and V^-, such that no arc exists between any two vertices within the same set. As such, a node of a bipartite graph only has neighbors in the other set. For a given matrix $\mathbf{X}$, denote by $\mathrm{Row}(\mathbf{X})$ and $\mathrm{Col}(\mathbf{X})$ its row set and column set, respectively, i.e., $\mathbf{X} = (x_{ij}|i \in \mathrm{Row}(\mathbf{X}), j \in \mathrm{Col}(\mathbf{X}))$, where x_{ij} is the (i,j)-entry. A bipartite graph on $\mathbf{X}$ is a graph $G(V^+, V^-, A)$ with a vertex set $V^+ = \mathrm{Row}(\mathbf{X})$ and $V^- = \mathrm{Col}(\mathbf{X})$ and the arc set $A = \{(i,j)|x_{ij} \neq 0\}$. By definition, each arc has the initial vertex in $\mathrm{Row}(\mathbf{X})$ and the terminal vertex in $\mathrm{Col}(\mathbf{X})$. Figure 5(b) shows the bipartite graph representation of the design matrix in Figure 5(a).

The decomposition procedure first needs to identify the *separating set* of the bigraph, which intuitively corresponds to the border rows in (2.2). In a graph, the separating set S is defined as the set of vertices, if there exist two vertices $a, b \notin S$,

such that all paths between a and b pass through at least one vertex of S [50]. Based on Menger's theorem [51], Even provided a detailed algorithm to obtain the smallest separating set [50]. Menger's theorem and Even's algorithm is presented in Appendix A. In order to find the border rows (not border columns), we need to restrict the final separating S to be a subset of Row($\mathbf{X}$), but directly applying Even's algorithm [50] does not guarantee so. Hence, we developed a method to find border rows still using Even's algorithm [50]. Our method starts with reducing the bigraph to only the vertices corresponding to Row($\mathbf{X}$). In doing so, we add a new arc between any two vertices in Row($\mathbf{X}$) that are connected through a vertex in Col($\mathbf{X}$), and subsequently, delete the vertices in Col($\mathbf{X}$) and their associated arcs. Figure 5(c) shows a reduced graph made from the bigraph in Figure 5(b). Then, we can apply Even's algorithm on the reduced graph to find S that is a subset of Row($\mathbf{X}$).

Once the separating set S, which corresponds to the border rows, is identified, the next step is to partition the rest of the matrix into disjoint submatrices. Denote by $\mathbf{X}[I, J]$ the submatrix of $\mathbf{X}$ with the row set I and the column set J, namely, $\mathbf{X}[I, J]$=$(x_{ij}|i \in I, j \in J)$; also let R = Row($\mathbf{X}$) and C = Col($\mathbf{X}$). The notation $\mathbf{X}[R - S, C]$ represents the rest of the original $\mathbf{X}$ matrix after a separating set S is removed. To identify each disjoint submatrix of $\mathbf{X}[R - S, C]$, one can apply the depths-first search (DFS) algorithm to the bipartite graph $G(R-S, C, A_r)$, where A_r denotes the arc set $\{(i, j)|x_{ij} \neq 0, i \notin S\}$; a detailed procedure of the DFS algorithm is in [50].

Combining the above two steps, the procedure of identifying the structure of a design matrix is summarized as follows.

1. For all $v_1, v_2 \in$ Row($\mathbf{X}$) and $v_1 \neq v_2$, if there exists $v_3 \in$ Col($\mathbf{X}$) such that v_1 and v_2 are connected through v_3, make a new arc connecting v_1 and v_2.

2. Delete all the nodes corresponding to Col($\mathbf{X}$) and their associated arcs. Then, we have a reduced graph only with the vertices corresponding to Row($\mathbf{X}$).

3. Using Even's algorithm [50], find the smallest separating set Q_1 of the reduced graph from Step 2.

4. Find $v_c \in \text{Col}(\mathbf{X})$, which has the least number of neighbors in the bipartite graph. Let $Q_2 \subseteq \text{Row}(\mathbf{X})$ be the set of all the neighbors of v_c.

5. If $|Q_1| \leq |Q_2|$, then $S = Q_1$. Otherwise $S = Q_2$.

6. Run DFS on $G(\text{Row}(\mathbf{X}) - S, \text{Col}(\mathbf{X}), A_r)$ to find the blocks.

In the above procedure, after applying Even's algorithm [50] in Step 3, the resulting smallest separating set of the reduced graph is generally the smallest separating subset of Row($\mathbf{X}$) for the original bigraph. However, there could exist an exception, which is captured by Step 4 and Step 5. By the definition of a bigraph, Q_2, a subset of Row($\mathbf{X}$) is also a separating set. The size of Q_2 could be smaller than that of Q_1. When that happens, Step 5 simply selects the smaller one between Q_1 and Q_2. It is not difficult to prove that in Step 5, S is the smallest separating subset of Row($\mathbf{X}$) for the bipartite graph.

Consider the design matrix in Figure 5 as an example. Using the above procedure, we first identify the sixth row as the separating set, which is also the border row. After removing the sixth row, the DFS algorithm will decompose the rest of the bipartite graph into two disconnected subgraphs, as illustrated in Figure 6. The two disconnected subgraphs indicate that the disjoint submatrices are $\mathbf{X}[\{2,3,5\},\{1,2,3\}]$ and $\mathbf{X}[\{1,4\},\{4\}]$, i.e., the blocks in the matrix in a bordered block form.

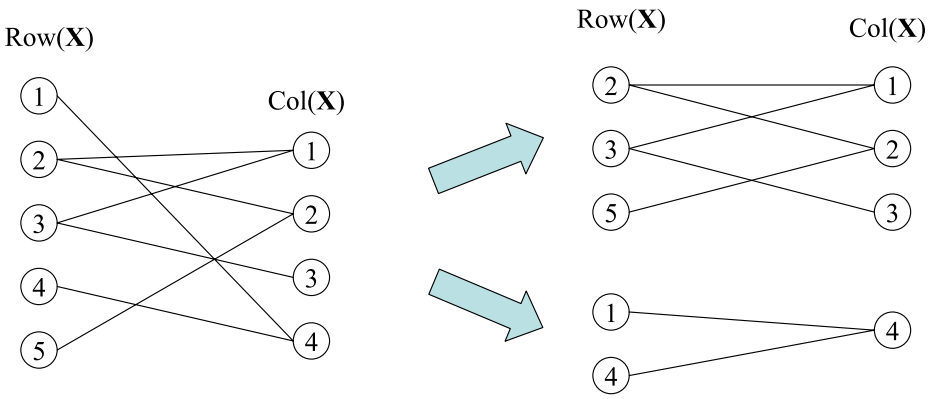


Fig. 6. The identification of blocks by DFS on the bipartite graph

D. Summary

This chapter presents the connection between a cluster pattern in a sensor network and Model (1.1); a design matrix of a clustered sensor network can be transformed into a bordered block form. For a design matrix in a bordered block form, border rows represent between-cluster links/communications, blocks represent within-cluster links/communications.

Section II.B and II.C present algorithms to identify a bordered block form based on hypergraph and reduced-reduced graph, respectively. Hypergraph based algorithms are broadly available [48, 49] but require users to specify some parameter values which may not be straightforward to use in finding a bordered block form. The reduced graph based algorithm presented in this dissertation does not require users to specify any parameter values.

CHAPTER III

COMPUTING SENSOR REDUNDANCY DEGREE: MATROID THEORY

As aforementioned in Chapter II, cluster patterns in a sensor network allow us to decompose a large-size sensor network into smaller subsystems of which the redundancy degree can be computed more efficiently. Presented in this chapter, are the details about such algorithms derived from matroid theory.

This chapter is organized as follows. First, matroid theory is briefly introduced in Section III.A and III.B. After that, theorems and lemmas related to the algorithms for the redundancy degree with their proofs are presented in Section III.C. The algorithms for the redundancy degree are presented in Section III.D, and the algorithm for a lower bound of the redundancy degree is presented in Section III.E.

A. Basic concepts in matroid theory

Matroid theory started from the algebraic theory of linear dependence and has been employed in many areas such as graph theory, lattice theory, electrical system theory, scheduling, and linear programming. The abstract properties in matroid theory enables various problems, which are equivalent in a matroid framework, to be solved by general matroid-based algorithms. Also, matroid theory suggests a profound framework that further enhances and integrates existing specialized algorithms. In this dissertation, the author utilizes matroid theory to devise algorithms for finding the cogirth of a vector matroid which is equivalent to finding the redundancy degree.

This dissertation mostly follows Oxley [52] in matroid terminology. For thorough introduction on matroid theory, please see [52] or [53]. Let M be a matroid $(E, \mathcal{I})$ consisting of a ground set E and a collection $\mathcal{I}$ of independent subsets of E. M satisfies so-called *independence augmentation axiom*; i.e., if I_1 and I_2 are in $\mathcal{I}$ and

$|I_1| < |I_2|$, then there is an element e of $I_2 - I_1$ such that $I_1 \cup e \in \mathcal{I}$. One example of a matroid is a *vector matroid*, $M[\mathbf{A}]$, which is obtained from a matrix $\mathbf{A}$, where E is the set of column label of $\mathbf{A}$, and $\mathcal{I}$ is the set of subsets of E such that the subsets are linearly independent. Suppose $A \subseteq E$; then, the rank of A, $r(A) = \max\{|I| : I \text{ is an indepedent subset of } A\}$. A *base* is a maximal independent set, and by $\mathcal{B}(M)$ we denote the collection of bases of M. A *spanning set* is a subset of E that contains a base. A *hyperplane* is a maximal non-spanning set. The *dual* matroid M^* has the bases $\mathcal{B}(M^*) = \{E - B : B \in \mathcal{B}(M)\}$. A *circuit* is a minimal dependent set, and by $\mathcal{C}(M)$ we denote the collection of circuits of M. We call $C \in \mathcal{C}(M^*)$ as a *cocircuit*. $\mathcal{B}(M^*)$ and $\mathcal{C}(M^*)$ is also denoted by $\mathcal{B}^*(M)$ and $\mathcal{C}^*(M)$, respectively. The *girth* of a matroid M, $g(M)$ is the cardinality of the smallest circuit in M and the *cogirth* $g^*(M)$ is the cardinality of the smallest cocircuit in M.

The redundancy degree $\eta(\mathbf{X})$ and the cogirth $g^*(M)$ has following relationship.

Proposition 1 *For a given matrix* $\mathbf{X}$*,*

$$\eta(\mathbf{X}) = g^*(M[\mathbf{X}^T]) - 1,$$

where $\mathbf{X}^T$ *is a transposed matrix of* $\mathbf{X}$*.*

Proof. For the proof of Proposition 1, we need Lemma 2 proved on page 70 in [52].

Lemma 2 *Let* M *be a matroid on a ground set* E*. Suppose* $H \subseteq E$*. Then,* H *is a hyperplane if and only if* $E - H$ *is a cocircuit.*

Let H be a hyperplane of $M[\mathbf{X}^T]$ and $C = \text{Col}(\mathbf{X})$. By the definition of a hyperplane, $\text{rank}(\mathbf{X}[H, C]) = p - 1$. Therefore, for $d \geq n - \max|H|$, there exists $\mathbf{X}_{(-d)}$ such that $\text{rank}(\mathbf{X}_{(-d)}) < p$. By Lemma 2, $n - \max|H|$ is equivalent to the minimum size of the cocircuits of $M[\mathbf{X}^T]$. Denote by D a cocircuit of $M[\mathbf{X}^T]$. We can conclude that if $d \geq \min|D|$, there exist $\mathbf{X}_{(-d)}$ such that $\text{rank}(\mathbf{X}_{(-d)}) < p$.

Suppose $T \subseteq E$ such that $|T| = \max|H| + 1$. Then, T is a spanning set, and thus, $\text{rank}(\mathbf{X}_{(-d)}) = p$ for $d \leq n - |T| = n - \max|H| - 1$. Since $\max|D| = n - \min|H|$, $\text{rank}(X_{(-d)}) = p$ for $d \leq \min|D| - 1$. Therefore, by the definition of the redundancy degree $\eta(\mathbf{X})$, we can conclude $\eta(\mathbf{X}) = \min|D| - 1$. $\square$

Proposition 1 reveals the equivalence of finding the redundancy degree and the cogirth of a vector matroid. For a graphic matroid, which can be regarded as a special form of a vector matroid, many efficient polynomial-time algorithms have been developed for finding the (co)girth. Denote a graph by $G = (V, E)$, where $V(G)$ is a non-empty set of vertices and $E(G)$ is a multiset of edges. The cocircuits of a graphic matroid $M(G)$ are the minimal subsets of $E(G)$ whose removal increases the number of components of G and are also called *cut-sets* of G. Suppose G is connected, the cogirth of $M(G)$ is the *edge-connectivity* of G [54]. The edge-connectivity received attentions from many researchers because it is used to measure how reliable a graph is. For example, Matula [55] developed an algorithm that finds the edge-connectivity and a minimum cut in $O(|V||E|)$ time; Nagamochi and Ibaraki [56] have developed an alternative algorithm to find the edge-connectivity in $O(|E| + \lambda(G)|V|^2)$ time, which is at least as good as the $O(|V||E|)$ time bound, where $\lambda(G)$ is the edge-connectivity.

Unfortunately, the cogirth problem for a more general matroid (e.g., a vector matroid) is a challenging problem. In 1971, Welsh [57] called for an efficient algorithm for finding the shortest circuit of a vector matroid over a field $\mathbb{F}$, but no polynomial-time algorithm has been found. Vardy [58] later proved that the minimum distance problem in binary coding, which can be converted to the girth problem of a vector matroid over a field $\mathbb{GF}(2)$, is NP-complete. The NP-completeness of the minimum distance problem implies that a polynomial-time algorithm for the cogirth of a vector matroid as well as of a general matroid is unlikely to exist [58].

Probably because of this theoretical difficulty, there are not many existing meth-

Algorithm 1 *Computing the redundancy degree* $\eta(\mathbf{X})$ *(i.e.,* $g^*(M[\mathbf{X}^T]) - 1$*)*

Input: matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$

Set $d = 1$*;*

Loop

  While $d \leq n - p + 1$

    If there exist $\mathbf{X}_{(-d)}$ *such that* $r(\mathbf{X}_{(-d)}) < r(\mathbf{X})$

      $\eta(\mathbf{X}) = d - 1$ *and stop;*

    Set $d = d + 1$*;*

ods to find the cogirth of a vector matroid; for instance, Mili and Coakley [41] did not specify how to find their cogirth-equivalent quantity in their D-estimator. To the best of our knowledge, two existing alternatives for finding the cogirth are an exhaustive rank testing procedure [11] and the circuit enumeration algorithm [59]. The exhaustive rank testing that finds the cogirth of a vector matroid is summarized as Algorithm 1. Throughout the paper, denote by $\mathbf{X}_{(-d)}$ the reduced matrix after deleting its d rows and by $r(\mathbf{X})$ the rank of $\mathbf{X}$.

The limitation of this exhaustive rank testing algorithm is that it may run into heavy computation when $g^*(M)$ is large, because the fourth line ("If there exist ...") in Algorithm 1 takes $\binom{m}{d}$ iterations; the computation time is proportional to $\sum_{d=1}^{g^*(M)} \binom{m}{d}$.

The circuit enumeration [59] and hyperplane generation [60] algorithms, which are proven to be polynomial-rate enumeration algorithms, provide alternative ways for finding the cogirth. In large-size systems, when the number of cocircuits is relatively large, these enumeration algorithms may not be practical. In fact, enumerating all circuits (or hyperplanes) is unnecessary if one only wants to find the cogirth.

B. Matroid connectivity

Before we present and prove properties about the (co)girth of a connected matroid in Section III.C, we introduce the concept of matroid connectivity. Matroid connectivity has been defined in [52] using matroid restriction and deletion. In order to facilitate the later derivations and discussions, we briefly review the concepts regarding restriction and deletion. After that, we review the connectivity of a matroid.

1. Restriction and deletion

Let M be the matroid $(E, \mathcal{I})$ consisting of a ground set E and a collection $\mathcal{I}$ of independent subsets of E. Suppose that $A \subseteq E$ and $I \in \mathcal{I}$, Let $\mathcal{I}|A$ be $\{I \subseteq A : I \in \mathcal{I}\}$. Then $(A, \mathcal{I}|A)$ is a matroid called the *restriction of M to A* or the *deletion of $E - A$ from M*. It is denoted by $M|A$ or $M\backslash(E - A)$.

Suppose $S \subseteq E(G)$, then $G\backslash S$ denotes the graph obtained from G by deleting the edges in S. We can easily see that

$$M(G\backslash S) = M(G)\backslash S.$$

Let $\mathbf{A}$ be a matrix, and S be a subset of E, the set of column labels of $\mathbf{A}$. Let $\mathbf{A}\backslash S$ be the matrix obtained from $\mathbf{A}$ by deleting all the columns whose labels are in S. The following property can be found on page 112 in [52].

$$M[\mathbf{A}]\backslash S = M[\mathbf{A}\backslash S].$$

2. Connectivity of a matroid

The matroid M is disconnected if and only if, for some proper non-empty subset T of $E(M)$,

$$\mathcal{I}(M) = \{I_1 \cup I_2 : I_1 \in \mathcal{I}(M|T), I_2 \in \mathcal{I}(M|(E - T))\},$$

This property of a disconnected matroid implies that

$$r(M) = r(T) + r(E - T).$$

Since $r^*(T) = |T| - r(M) + r(E - T)$, where $r^*(T) = r((M|T)^*)$,

$$r(T) + r^*(T) - |T| = 0.$$

Naturally, this implies that connectivity is self-dual. Hence, M is connected if and only if M^* is connected.

Let $M_1, M_2, \ldots, M_k$ be matroids on disjoint ground sets $E_1, E_2, \ldots, E_k$, respectively. Also let $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k$ be the collections of independent sets in $M_1, M_2, \ldots, M_k$, respectively. Suppose $E = E_1 \cup E_2 \cup \ldots \cup E_k$ and $\mathcal{I} = \{I_1 \cup I_2 \cup \ldots \cup I_k : I_j \in \mathcal{I}_j \text{ for all } j \in \{1, 2, \ldots, k\}\}$, then $M = (E, \mathcal{I})$ is a matroid and denoted by $M_1 \oplus M_2 \oplus \ldots \oplus M_k$. We call $M_1, M_2, \ldots, M_k$ the *direct sum components of* M, and M is called the *direct sum* or *1-sum* of $M_1, M_2, \ldots, M_k$,

The direct sum has following properties which are straightforward to prove.

- $\mathcal{C}(M_1 \oplus M_2 \oplus \ldots \oplus M_k) = \mathcal{C}(M_1) \cup \mathcal{C}(M_2) \cup \ldots \cup \mathcal{C}(M_k)$
- $(M_1 \oplus M_2 \oplus \ldots \oplus M_k)^* = M_1^* \oplus M_2^* \oplus \ldots \oplus M_k^*$

From these two properties, we can conclude that

$$\mathcal{C}^*(M_1 \oplus M_2 \oplus \ldots \oplus M_k) = \mathcal{C}^*(M_1) \cup \mathcal{C}^*(M_2) \cup \ldots \cup \mathcal{C}^*(M_k) \tag{3.1}$$

Hence, (co)girth of a disconnected matroid is the minimum of the (co)girths of its direct sum components.

Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be matrices over field $\mathbb{F}$, and $M_1 = M[\mathbf{A}_1]$ and $M_2 = M[\mathbf{A}_2]$. The direct sum $M = M_1 \oplus M_2$ can be regarded as a matroid on a matrix $\mathbf{A}$ over field

$\mathbb{F}$ in a block form such as (please refer to (2.1) in Chapter II)

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{pmatrix}. \tag{3.2}$$

From Equation (3.1), we can find the (co)girth of $M[\mathbf{A}]$ from the minimum of those of M_1 and M_2.

C. Cogirth of a connected matroid

Let M be a connected matroid. Suppose there exists $S \subset E(M)$ such that $M\backslash S$ is disconnected and $M_1, M_2, \ldots, M_k$ are the direct sum components of $M\backslash S$. Then, $M\backslash S = M_1 \oplus M_2 \oplus \ldots \oplus M_k$.

For $i = 1, \ldots, k$, we define

- $\mathcal{C}_i^*(M) = \{D \in \mathcal{C}^*(M) : D \subseteq E(M_i) \cup S\}$, and
- $c_i^*(M) = c_i^* = \min\{|D| : D \in \mathcal{C}_i^*(M)\}$.

Please note that $\mathcal{C}_i^*(M)$ is *not* equivalent to $\mathcal{C}^*(M_i)$.

The subsequent lemmas and theorems characterize the cogirth of a connected matroid. First, we present a result that will be used in the proofs of other lemmas and theorems; its proof is omitted because this is a straightforward result from the independence augmentation axiom in Section III.A.

Lemma 3 *Let I be an independent set in a matroid M. There exists a base B containing I in M.*

Lemmas 3 and 4 characterize the cogirth of a matroid when the restriction of the matroid can be decomposed into two direct sum components.

Lemma 4 *Suppose* $M\backslash S = M_1 \oplus M_2$. *Let* D *be a cocircuit in* M, *which is not in* $\mathcal{C}_1^*(M)$ *or* $\mathcal{C}_2^*(M)$. *Then,*

$$|D| \geq \max(c_1^* - |S|, 1) + \max(c_2^* - |S|, 1).$$

Proof. Let $P_1 = D \cap E(M_1)$, $P_2 = D \cap E(M_2)$, and $T = D \cap S$. Obviously, $D = P_1 \cup P_2 \cup T$. Because D is not in $\mathcal{C}_1^*(M)$ or $\mathcal{C}_2^*(M)$, P_1 and P_2 are not empty sets. First, we will show that $P_1 \cup S$ and $P_2 \cup S$ are codependent sets in M.

Assuming $P_1 \cup S$ is a coindependent set in M, there exists a base $B \subseteq E(M) - (P_1 \cup S) = E(M\backslash S) - P_1$ in M. Since $B \subseteq E(M\backslash S)$, B is also a base of $M\backslash S = M_1 \oplus M_2$. Then, there exists a base B_1 in M_1, and a base B_2 in M_2 such that $B = B_1 \cup B_2$. As D is a cocircuit and P_1 is not an empty set (D is not in $\mathcal{C}_2^*$), $P_2 \cup T$ is a coindependent set. Therefore, $r(E(M) - (P_2 \cup T)) = r(M)$. Since B_1 is an independent set in M_1, B_1 is also an independent set in $M\backslash(P_2 \cup T)$. By Lemma 3, there is a base B' in $M\backslash(P_2 \cup T)$ such that $B_1 \subseteq B'$. Since $r(M) = r(M\backslash(P_2 \cup T))$, B' is also a base in M.

Suppose $P_1 \cap B' \neq \emptyset$. Then, there exists $e \in P_1 \cap B'$. Since $B_1 \subseteq E(M_1) - P_1$, e is not in B_1. However, $e \in E(M_1)$, and B_1 is a base in M_1. Therefore, $\{e\} \cup B_1$ should be a dependent set in M_1. This contradicts the assumption that B' is a base in M, and thus, $P_1 \cap B' = \emptyset$. This implies that $B' \subseteq E - (P_1 \cup P_2 \cup T) = E - D$, which contradicts the assumption that D is a cocircuit. Therefore, $P_1 \cup S$ is a codependent set. Likewise, we can prove $P_2 \cup S$ is a codependent set.

Because $P_1 \cup S$ is a codependent set, $|P_1 \cup S| \geq c_1^*$, or

$$|P_1| \geq c_1^* - |S|.$$

Since $|P_1| \geq 1$,

$$|P_1| \geq \max(c_1^* - |S|, 1).$$

Likewise, it can be proven that $|P_2| \geq \max(c_2^* - |S|, 1)$. Therefore,

$$|P_1| + |P_2| \geq \max(c_1^* - |S|, 1) + \max(c_2^* - |S|, 1).$$

Thus, $|D| = |P_1 \cup P_2 \cup T| \geq \max(c_1^* - |S|, 1) + \max(c_2^* - |S|, 1)$. $\square$

Lemma 5 *If $M \backslash S = M_1 \oplus M_2$, and $g^*(M) \geq 2|S| - 1$, then*

$$g^*(M) = \min(c_1^*, c_2^*).$$

Proof. Let D be a cocircuit in M, which is not in $\mathcal{C}_1^*(M)$ or $\mathcal{C}_2^*(M)$. What we need to prove is

$$|D| \geq \min(c_1^*, c_2^*). \tag{3.3}$$

From Lemma 4,

$$|D| \geq \max(c_1^* - |S|, 1) + \max(c_2^* - |S|, 1).$$

Since $\max(c_1^* - |S|, 1) + \max(c_2^* - |S|, 1) \geq c_1^* + c_2^* - 2|S|$,

$$|D| \geq c_1^* + c_2^* - 2|S| \geq 2\min(c_1^*, c_2^*) - 2|S|.$$

Because $|D| \geq 2|S| - 1$,

$$|D| \geq 2\min(c_1^*, c_2^*) - |D| - 1,$$

or

$$|D| \geq \min(c_1^*, c_2^*) - \frac{1}{2}.$$

Therefore,

$$|D| \geq \min(c_1^*, c_2^*). \quad \square$$

Lemma 5 can be extended to general cases with $M_1, M_2, \ldots, M_n$, as stated in Theorem 6, and it can be easily proven using Lemma 5.

Theorem 6 *If* $M\backslash S = M_1 \oplus M_2 \oplus \ldots \oplus M_k$ *and* $g^*(M) \geq 2|S| - 1$, *then*

$$g^*(M) = \min_{i \in \{1,2,\ldots,k\}} c_i^*.$$

Theorem 6 tells us that the cogirth of the connected matroid M can be obtained from $c_1^*, c_2^*, \ldots, c_k^*$ once the cogirth is known to be larger than the bound. The bound in the above results is specified as two times the cardinality of S subtracted by one. The following results will further relax this bound.

For $J \subseteq \{1, \ldots, k\}$, we define

- $\mathcal{C}_J^*(M) = \{D \in \mathcal{C}^*(M) : D \subseteq \bigcup_{j \in J} E(M_j) \cup S\}$, and
- $c_J^* = \min\{|D| : D \in \mathcal{C}_J^*\}$.

A *q-subset* is a subset of a set on n elements containing exactly q elements. The number of q-subsets on n elements is $\binom{n}{q}$. Let $\mathcal{J}_q$ be a collection of all the q-subsets of $\{1, 2, \ldots, k\}$.

Lemma 7 *Let* $Y \in \mathcal{J}_q$. *If* $M\backslash S = M_1 \oplus M_2 \oplus \ldots \oplus M_k$, *and* $c_Y^* \geq \frac{q}{q-1}|S| - 1$, *then*

$$c_Y^* = \min_{y \in Y} c_{Y-\{y\}}^*.$$

Proof. Let $D \in \mathcal{C}_Y^*(M)$, $P_i = D \cap E(M_i)$ for $i \in Y$, and $T = D \cap S$.

Our proof is divided into two cases. First, we will prove for the case that there exists $i \in Y$ such that $|P_i| = 0$. Second, we will prove the other case that every P_i for $i \in Y$ is not an empty set.

1. If there exists i such that P_i is an empty set, then $D \in \mathcal{C}_{Y-\{i\}}^*$. Then,

$$|D| \geq c_{Y-\{i\}}^* \geq \min_{y \in Y} c_{Y-\{y\}}^*. \tag{3.4}$$

2. When all P_i for $i \in Y$ is not an empty set; like the proof for Lemma 5, it can be shown that $P_i \cup S$ is a codependent set in M for $i \in Y$. Hence, for $y \in Y$,

$$\sum_{i \in Y-\{y\}} |P_i| \geq \sum_{i \in Y-\{y\}} c^*_i - (q-1)|S|.$$

Since $\sum_{i \in Y-\{y\}} c^*_i \geq (q-1)c^*_{Y-\{y\}}$,

$$\sum_{i \in Y-\{y\}} |P_i| \geq (q-1)c^*_{Y-\{y\}} - (q-1)|S|.$$

Since $\sum_{y \in Y} c^*_{Y-\{y\}} \geq q \min_{y \in Y} c^*_{Y-\{y\}}$, adding up for all $y \in Y$,

$$\sum_{y \in Y} \sum_{i \in Y-\{y\}} |P_i| \geq q(q-1) \min_{y \in Y} c^*_{Y-\{y\}} - q(q-1)|S|.$$

Since $\sum_{y \in Y} \sum_{i \in Y-\{y\}} |P_i| = (q-1) \sum_{i \in Y} |P_i| = (q-1)|D| - (q-1)|T|$,

$$|D| \geq q \min_{y \in Y} c^*_{Y-\{y\}} - q|S| + |T| \geq q \min_{y \in Y} c^*_{Y-\{y\}} - q|S|.$$

Because $q|S| \leq (q-1)|D| + (q-1)$,

$$|D| \geq q \min_{y \in Y} c^*_{Y-\{y\}} - (q-1)|D| - (q-1),$$

or

$$|D| \geq \min_{y \in Y} c^*_{Y-\{y\}} - \frac{(q-1)}{q}.$$

Since $c^*_Y = \min\{|D| : D \in \mathcal{C}^*_Y\}$,

$$c^*_Y \geq \min_{y \in Y} c^*_{Y-\{y\}}. \tag{3.5}$$

From equation (3.4) and (3.5),

$$c^*_Y \geq \min_{y \in Y} c^*_{Y-\{y\}}.$$

Since $\mathcal{C}^*_Y$ contains all the cocircuits in $\mathcal{C}^*_{Y-\{y\}}$ for all $y \in Y$,

$$c^*_Y = \min_{y \in Y} c^*_{Y-\{y\}}. \quad \square$$

Using Lemma 7, we prove Theorem 8, which generalizes Theorem 6. In fact, Theorem 6 is the special case of Theorem 8 when $q = 2$.

Theorem 8 *If $M\backslash S = M_1 \oplus M_2 \oplus \ldots \oplus M_k$, and $g^*(M) \geq \frac{q+1}{q}|S| - 1$, then*

$$g^*(M) = \min_{J \in \mathcal{J}_q} c^*_J.$$

Proof. Suppose $g^*(M) = c^*_{\{1,2,\ldots,k\}} \neq \min_{J \in \mathcal{J}_q} c^*_J$. Then, there exists q' such that $q+1 \leq q' \leq n$, and

$$\min_{Y \in \mathcal{J}_{q'}} c^*_Y \neq \min_{X \in \mathcal{J}_{q'-1}} c^*_X.$$

Since $q' \geq q+1$,

$$\min_{Y \in \mathcal{J}_{q'}} c^*_Y \geq g^*(M) \geq \frac{q+1}{q}|S| - 1 \geq \frac{q'}{q'-1}|S| - 1.$$

The condition in Lemma 5 is satisfied, so

$$\min_{Y \in \mathcal{J}_{q'}} c^*_Y = \min_{Y \in \mathcal{J}_{q'}} \{\min_{y \in Y} c^*_{Y-\{y\}}\} = \min_{X \in \mathcal{J}_{q'-1}} c^*_X,$$

which leaves us with a contradiction. Thus,

$$g^*(M) = c^*_{\{1,2,\ldots,n\}} = \min_{J \in \mathcal{J}_q} c^*_J. \quad \square$$

Theorems 6 and 8 provide the basis for our latter algorithms, which allows us to find the cogirth without having to enumerate all cocircuits when a certain bound condition is satisfied. The details of applying Theorems 6 and 8 will be explained in Section III.D.

D. Algorithms for finding redundancy degree (cogirth of a vector matroid)

Algorithms in this section aim at finding the redundancy degree $\eta(\mathbf{X})$, or equivalently cogirth of a vector matroid $g^*(\mathbf{X}^T)$, where $\mathbf{X}$ is in a bordered block form. Again, consider a $n \times p$ design matrix $\mathbf{X}$ in a bordered block form with k blocks. For details about finding a border block form, please refer to Chapter II.

Lemma 10 shows the relationship between $c_J^*(M)$ and the cogirth of a submatrix, where J is a subset of $\{1, 2, \ldots, k\}$. In order to prove Lemma 10, the following remark is derived using linear algebra.

Remark 9 *The cocircuit in* $M[\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]^T]$ *is a codependent set in* $M[\mathbf{X}^T]$.

Please note that $\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]$ is a submatrix of $\mathbf{X}$ containing border rows and blocks corresponding to J. Here, $\mathbf{X}[\cdot]^T$ is a transposed matrix of $\mathbf{X}[\cdot]$.

From Remark 9, we can conclude,

$$c_J^*(M[\mathbf{X}^T]) \leq c^*(M[\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]^T]). \tag{3.6}$$

Lemma 10 presents that $c_J^*(M)$ can be obtained from the cogirth of a submatrix associated with the blocks and the border when $r(M\backslash S) = r(M)$.

Lemma 10 *If* $r(M\backslash S) = r(M)$,

$$c_J^*(M[\mathbf{X}^T]) = c^*(M[\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]^T]),$$

where J *is a subset of* $\{1, 2, \ldots, k\}$.

Proof. Let $r = r(M)$ and $r_i = r(M_i)$ for $i = 1, 2, \ldots, k$. Since $r(M\backslash S) = r(M)$,

$$r = r_1 + r_2 + \ldots + r_k.$$

Let $D \in \mathcal{C}_J^*(M[\mathbf{X}^T])$. Then,

$$r(\mathbf{X}[R - D, C]) = r - 1.$$

Since $r(M\backslash S) = r(M)$,

$$r(\mathbf{X}[R - \bigcup_{i\in J} R_i \cup S, C - \bigcup_{j\in J} C_j]^T) = r - \sum_{j\in J} r_j.$$

Hence,

$$r(\mathbf{X}[\bigcup_{i\in J} R_i \cup S - D, \bigcup_{j\in J} C_j]^T) = \sum_{j\in J} r_j - 1$$

Therefore, the cocircuits in $\mathcal{C}_J^*(M[\mathbf{X}^T])$ is a codependent set in $M[\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]^T]$, and

$$c_J^*(M[\mathbf{X}^T]) \geq c^*(M[\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]^T]).$$

Combining with equation (3.6),

$$c_J^*(M[\mathbf{X}^T]) = c^*(M[\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j]^T]). \quad \square$$

As long as $g^*(M) > 1$, the bound condition in Theorem 6, $g^*(M) \geq 2|S| - 1$, implies that $g^*(M) > |S|$. From the definition of cogirth, $g^*(M) > |S|$ means that S is not a cocircuit, and thus, $r(M\backslash S) = r(M)$. This is equivalent to saying that when $g^*(M) > 1$ and the bound condition is also satisfied, the condition $r(M\backslash S) = r(M)$ will be satisfied. The following cogirth-finding algorithm is based on Theorem 6 and Lemma 10.

As a consequence of Theorem 8, the following corollary is derived to help establish Algorithm 3, which can further save the computation time by reducing the bound $(2|S| - 1)$ in the fourth step of Algorithm 2 to $(\frac{n}{n-1}|S| - 1)$.

Algorithm 2 *Computing the redundancy degree* $\eta(\mathbf{X})$ *(i.e.,* $g^*(M[\mathbf{X}^T]) - 1$*)*

Input: a calibration matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$*, the border rows* S *of* $\mathbf{X}$*, and the row set and column set of blocks* $(R_1, R_2, \ldots, R_k)$ *and* $(C_1, C_2, \ldots, C_k)$*.*

Set $d = 1$*;*

Loop

While $d \leq 2|S| - 2$

If there exist $\mathbf{X}_{(-d)}$ *such that* $r(\mathbf{X}_{(-d)}) < r(\mathbf{X})$

$\eta(\mathbf{X}) = d - 1$ *and stop;*

Set $d = d + 1$*;*

Loop

While $d \leq n - p + 1$

If there exists $\mathbf{X}^{(i)}_{(-d)}$ *such that* $r(\mathbf{X}^{(i)}_{(-d)}) < r(\mathbf{X}^{(i)})$ *for* $i \in \{1, 2, \ldots, k\}$

$\eta(\mathbf{X}) = d - 1$ *and stop;*

Set $d = d + 1$*;*

Corollary 11 *If* $M\backslash S = M_1 \oplus M_2 \oplus \ldots \oplus M_k$, *and* $g^*(M) \geq \frac{q+1}{q}|S| - 1$,

$$g^*(M) = \min_{J \in \mathcal{J}_q} c^*(M[\mathbf{X}[\bigcup_{i \in J} R_i \cup S, \bigcup_{j \in J} C_j]^T]).$$

Recall that $\mathcal{J}_q$ denotes the collection of all the q-subsets of $\{1, 2, \ldots, k\}$.

Proof. Suppose $|S| \neq 0$. (The case $|S| = 0$ is straightforward, so omitted.) Since $g^*(M) \geq \frac{q+1}{q}|S| - 1 > |S| - 1$,

$$g^*(M) \geq |S|.$$

Hence, strict subsets of S are not cocircuits in M.

1. If S is not a cocircuit, $r(M\backslash S) = r(M)$. By Theorem 8 and Lemma 10,

$$g^*(M) = \min_{J \in \mathcal{J}_q} c^*_J = \min_{J \in \mathcal{J}_q} c^*(M[\mathbf{X}[\bigcup_{i \in J} R_i \cup S, \bigcup_{j \in J} C_j]^T]). \tag{3.7}$$

2. If S is a cocircuit in M, then $g^*(M) = |S|$. Suppose S is coindependent set in $M[\mathbf{X}[R_i \cup S, C_i \cup S]]$ for all $i \in \{1, 2, \ldots, k\}$. Then,

$$\sum_{i=1}^{n} r(\mathbf{X}[R_i, C_i]) = r(\mathbf{X}).$$

   This contradicts the assumption that S is a cocircuit in M. Therefore, there exists $i \in \{1, 2, \ldots, n\}$ such that S is a codependent set in $M[\mathbf{X}[R_i \cup S, C_i]^T]$. By Remark 9, for $j \in \{1, 2, \ldots, k\}$, the sizes of the cocircuits in $M[\mathbf{X}[R_j \cup S, C_j]^T]$ are not smaller than $g^*(M)$. Therefore,

$$g^*(M) = \min_{i \in \{1,2,\ldots,k\}} c^*(M[\mathbf{X}[R_i \cup S, C_i]^T]). \tag{3.8}$$

From equations (3.7) and (3.8), we can conclude,

$$g^*(M) = \min_{J \in \mathcal{J}_q} c^*_J = \min_{J \in \mathcal{J}_q} c^*(M[\mathbf{X}[\bigcup_{i \in J} R_i \cup S, \bigcup_{j \in J} E_j]^T]). \quad \square$$

Corollary 11 enables us to establish Algorithm 3, which uses a smaller bound

($\frac{k}{k-1}|S|-1$) than the bound ($2|S|-1$) in Algorithm 2. When using the bound of ($\frac{k}{k-1}|S|-1$), instead of testing the rank of a submatrix $\mathbf{X}[R_i \cup S, C_i]$ generated from an individual block, we can test one of the possible submatrices $\mathbf{X}[\bigcup_{j\in J} R_j \cup S, \bigcup_{j\in J} C_j]$ that combine blocks. As specified in Theorem 8 and Corollary 11, J is an element in $\mathcal{J}_q$, which is the collection of all the q-subsets of $\{1, 2, \ldots, k\}$. We therefore need additional steps in the next algorithm to sort through the combinations of blocks for Corollary 11 to be applied.

In so doing, denote by $\mathcal{X}_q$ a collection of matrices such that

$$\mathbf{X}[\bigcup_{i\in J} R_i \cup S, \bigcup_{j\in J} C_j] \in \mathcal{X}_q \text{ for all } J \in \mathcal{J}_q.$$

Obviously, $\mathcal{X}_1$, i.e., $q = 1$, is the collection $\{\mathbf{X}[R_1 \cup S, C_1], \ldots, \mathbf{X}[R_n \cup S, C_n]\}$, and $\mathcal{X}_k$ (i.e., $q = k$) contains only $\mathbf{X}$. Denote by $N_{(-d)}(q)$ the total number of reduced matrices generated from *all* the matrices in $\mathcal{X}_q$, so

$$N_{(-d)}(q) = \sum_{J\in\mathcal{J}_q} \binom{|\bigcup_{j\in J} R_j| + |S|}{d}.$$

The algorithm using Corollary 11 is as follows.

At the fifth step, the algorithm selects q that requires the least computation time. The computation time for step (6) is proportional to the number of matrices $\mathbf{X}'_{(-d)}$ that can be generated by removing d rows from $\mathbf{X}' \in \mathcal{X}_k$. For a given d, we should choose q^* that minimizes $N_{(-d)}(q)$, namely $q^* = \arg\min_q N_{(-d)}(q)$, and then, test the ranks of the resulting reduced matrices. Note that for a given d, the possible value for q is bounded in the range $[\frac{|S|}{d-|S|+1}, n]$, as a direct result from the conditions specified in Theorem 8 and Corollary 11. If $|S|$ is considerably large, Algorithm 3 may become identical to Algorithm 1, and will not gain any additional computation benefit. When there is no border or a just one-column border in $\mathbf{X}$, $2|S| - 1 < 1$.

Algorithm 3 *Computing the redundancy degree* $\eta(\mathbf{X})$ *(i.e.,* $g^*(M[\mathbf{X}^T]) - 1$*)*

Parameters: integer $d \geq 1$.

Input: a calibration matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$*, the border rows* S *of* $\mathbf{X}$*, and the row set and column set of blocks* $(R_1, R_2, \ldots, R_k)$ *and* $(C_1, C_2, \ldots, C_k)$.

Set $d = 1$*;*

Loop

While $d \leq \frac{k}{k-1}|S| - 2$

If there exist $\mathbf{X}_{(-d)}$ *such that* $r(\mathbf{X}_{(-d)}) < r(\mathbf{X})$

$\eta(\mathbf{X}) = d - 1$ *and stop;*

Set $d = d + 1$*;*

Loop

While $d \leq n - p + 1$

Find q^* *such that*

$$q^* = \arg\min_{\frac{|S|}{d-|S|+1} \leq q \leq n} N_{(-d)}(q).$$

For $\mathbf{X}' \in \mathcal{X}_{q^*}$*, if there exists* $\mathbf{X}'_{(-d)}$ *such that* $r(\mathbf{X}'_{(-d)}) < r(\mathbf{X}')$

$\eta(\mathbf{X}) = d - 1$ *and stop;*

Set $d = d + 1$*;*

As a result, Algorithm 3 will reap the greatest computation benefit by testing those small matrices in $\mathcal{X}_1$ except when $d = 1$. For any other cases, the computation saving lies in between.

E. Algorithm for finding a lower bound of redundancy degree

Algorithm 2 and 3 work very efficiently for a matrix with a small $|S|$ and small-sized blocks, but they are not so efficient when the size of $\mathbf{X}$ or $|S|$ is large. The problem is that the first loop of Algorithm 2 or 3, which goes until $d < 2|S| - 1$ or $d < \frac{k}{k-1}|S| - 1$, may take too much computation time, since it tests the ranks of the original matrix. Under that circumstance, computing the exact degree of redundancy degree may become unaffordable.

It turns out that a lower bound of the redundancy degree is valuable for robust estimation in sensor network applications; we will see more about the benefit of using the lower bound of the redundancy degree in Chapter IV. Also, the benefit by using the lower bound is illustrated using a wireless sensor network in Chapter V.

Searching for a lower bound of the redundancy degree is usually much easier than for the exact redundancy degree. Algorithm 4 presented in the latter part of this section computes a lower bound of the redundancy degree for a large-sized $\mathbf{X}$, a large $|S|$, or both. The computation benefit of Algorithm 4 comes from that it tests merely the ranks of sub-matrices and avoids testing the original matrix.

Lemma 4 suggests a lower bound of $\eta(\mathbf{X})$ when $M[\mathbf{X}^T]/S = M[\mathbf{X}[R_1, C_1]^T] \oplus M[\mathbf{X}[R_2, C_2]^T]$. We rewrite Lemma 4 in terms of redundancy degree and the lower bound of the redundancy degree as Remark 12. For simplicity, let's denote by $\mathbf{X}^{(1)} = \mathbf{X}[R_1 \cup S, C_1]$ and by $\mathbf{X}^{(2)} = \mathbf{X}[R_2 \cup S, C_2]$ and call them *cluster matrices*.

Remark 12 *Define* $l(\mathbf{X}) = \max\{\eta(\mathbf{X}^{(1)}) - |S|, 0\} + \max\{\eta(\mathbf{X}^{(2)}) - |S|, 0\} + 1$.

1. *If* $l(\mathbf{X}) \geq \min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)})\}$,

$$\eta(\mathbf{X}) = \min\{\eta(\mathbf{X}[R_1 \cup S, C_1]), \eta(\mathbf{X}[R_2 \cup S, C_2])\}.$$

2. *If* $l(\mathbf{X}) < \min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)})\}$,

$$\eta(\mathbf{X}) \geq l(\mathbf{X}).$$

In order to better reflect the lower bound aspect, the following corollary is stated as a direct result from Remark 12. Note that computing $\min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)}), l(\mathbf{X})\}$ does not involve $\mathbf{X}$ directly but involves cluster matrices $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$. So computing the lower bound becomes easier.

Corollary 13 $\eta(\mathbf{X}) \geq \min\{\eta(\mathbf{X}^{(1)}), \eta(\mathbf{X}^{(2)}), l(\mathbf{X})\}$.

Because of the structural result stated in Lemma 4, Algorithm 4 can recursively decompose the matrix. The following mathematical derivation is for a recursive procedure to compute the lower bound $\nu(\mathbf{X})$ of the redundancy degree (i.e., $\nu(\mathbf{X}) \leq \eta(\mathbf{X})$). Apparently,

$$\eta(\mathbf{X}^{(1)}) \geq \nu(\mathbf{X}^{(1)}), \text{ and } \eta(\mathbf{X}^{(2)}) \geq \nu(\mathbf{X}^{(2)}). \tag{3.9}$$

Thus,

$$l(\mathbf{X}) \geq \max\{\nu(\mathbf{X}^{(1)}) - |S|, 0\} + \max\{\nu(\mathbf{X}^{(2)}) - |S|, 0\} + 1. \tag{3.10}$$

Using (3.9) and (3.10), it is straightforward to get the following result from Corollary 13.

Corollary 14 *Define* $l_\nu(\mathbf{X}) = \max\{\nu(\mathbf{X}^{(1)}) - |S|, 0\} + \max\{\nu(\mathbf{X}^{(2)}) - |S|, 0\} + 1$; *then,*

$$\eta(\mathbf{X}) \geq \min\{\nu(\mathbf{X}^{(1)}), \nu(\mathbf{X}^{(2)}), l_\nu(\mathbf{X})\}.$$

Corollary 14 benefits the computation because $\nu(\mathbf{X}^{(1)})$ and $\nu(\mathbf{X}^{(2)})$ only evaluate a second-layer (and thus smaller) cluster matrices embodied in a first-layer cluster matrix. This action can be applied to the next layer until a cluster matrix can no longer be decomposed. As the decomposition goes deep, the computation becomes less and less demanding; on the other hand, the lower bound found also becomes looser, which is obvious by noting the two inequalities used in (3.9) and (3.10). Using a very loose lower bound will eventually make a robust estimator no different from an ordinary LS estimator. Therefore, for the sake of robust estimation, there is a tradeoff between the computation consideration and the robustness consideration. A simple rule of thumb is to find the tightest lower bound of which the computation is tolerable to one's current resource.

We also make an extension from Theorem 6 as Corollary 15, which is a straightforward result from equation (3.9) and Theorem 6.

Corollary 15 *If* $\eta(\mathbf{X}) \geq 2|S| - 2$, *then*

$$\eta(\mathbf{X}) \geq \min\{\nu(\mathbf{X}^{(1)}), \nu(\mathbf{X}^{(2)})\}.$$

Finally, we present the following recursive Algorithm 4 based on both Corollary 14 and 15. In addition to $\mathbf{X}$, Algorithm 4 needs a constant K, which decides the condition of which corollary to invoke. This constant K sets a threshold of computation load for Algorithm 4. Using a small K may reduce the computation time, but may result in a poor lower bound that is far from the actual $\eta(\mathbf{X})$. On the contrary, using a large K could lose the computation benefit of Algorithm 4 even though the lower bound found may be close to the actual $\eta(\mathbf{X})$. To that extent, K should be chosen appropriately meeting the computing requirement of a problem. We select K to be one million for a computer with 3.6GHz Pentium CPU and 4G memory in examples in Section 4.

Algorithm 4 *Computing the lower bound of* $\eta(\mathbf{X})$

Input: $\mathbf{X} \in \mathbb{R}^{n \times p}$ *and a constant* K*.*

Function: Lowerbound($\mathbf{Z}$*,* d*)* {

Find the border rows S *and the cluster matrices of* $\mathbf{Z}$ *(i.e.,* $\mathbf{Z}^{(1)}$ *and* $\mathbf{Z}^{(2)}$*).*

If $|S| \geq |C|$ *or* $|S| = \min(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)})$

Run the enumerative rank testing algorithm starting from d*, and return* $\eta(\mathbf{Z})$*;*

Else if $\binom{|R|}{2|S|-2} \geq K$ *%If* $|R|$ *and/or* $|S|$ *are large*

Set $l_1 = Lowerbound(\mathbf{Z}^{(1)}, d)$ *%Finding* $\nu(\mathbf{Z}^{(1)})$

Set $l_2 = Lowerbound(\mathbf{Z}^{(2)}, d)$ *%Finding* $\nu(\mathbf{Z}^{(2)})$

Return $\min\{l_1, l_2, \max\{l_1 - |S|, 0\} + \max\{l_2 - |S|, 0\} + 1\}$*; %Corollary 14*

Else

Loop

While $d \leq 2|S| - 2$

If there exist $\mathbf{Z}_{(-d)}$ *such that* $r(\mathbf{Z}_{(-d)}) < r(\mathbf{Z})$

Return $d - 1$*;*

Set $d = d + 1$*;*

Return $\min\{Lowerbound(\mathbf{Z}^{(1)}, d), Lowerbound(\mathbf{Z}^{(2)}, d)\}$*; %Corollary 15*

}

Run Lowerbound($\mathbf{X}$*, 1)*

F. Summary

This chapter provides new algorithms to compute, using a matroid decomposition, the redundancy degree, which is equivalent to the cogirth of a vector matroid. For the decomposition of a connected matroid, Theorem 6 and Theorem 8 have been derived based upon matroid duality and connectivity. Lemma 10 and Corollary 11 lead to developing Algorithm 2 and Algorithm 3 for vector matroids. To characterize a vector matroid connectivity, we used a border block form of a matrix, explained in Chapter II.

Algorithm 2 and 3 in Section III.D work very efficiently for a matrix having a small $|S|$ and small-sized blocks, but may still take too much computation when the size of $\mathbf{X}$ or $|S|$ is large. Section III.E provides an algorithm to compute a lower bound of the redundancy degree that is useful for sensor network applications; algorithm 4 computes a lower bound of the redundancy degree for a large-sized $\mathbf{X}$, a large $|S|$, or both. The computation benefit of Algorithm 4 comes from merely testing the ranks of sub-matrices.

CHAPTER IV

MEASURING ROBUSTNESS OF SENSOR NETWORKS

In this chapter, we connect the redundancy degree $\eta(\mathbf{X})$ to a robustness measure - the breakdown point and fault tolerance capability - and introduce a robust estimation procedure that attains the maximum breakdown point or fault tolerance capability.

Chapter IV is organized as follows: Section IV.A introduces background about robust regression and breakdown point. In Section IV.B, we define the fault tolerance capability using the breakdown point. Section IV.C includes more discussion about the LTS estimation, and Section IV.D presents LTS estimation using a lower bound of the redundancy degree that may not attain the maximum fault tolerance capability but attains a certain level of the fault tolerance capability.

A. Robust regression and breakdown point

In Model (1.1), the error term $\mathbf{e}$ is typically assumed to be normally distributed with zero mean and covariance matrix $\sigma^2\mathbf{I}$. To estimate $\boldsymbol{\beta}$, the most commonly used method is the least squares (LS) estimator, that is, $\hat{\boldsymbol{\beta}}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. One major drawback of the LS estimator is its lack of robustness; heavy-tailed distributions in $\mathbf{e}$ and outliers in $\mathbf{y}$ (called y-outlier) and/or $\mathbf{X}$ (called x-outlier) prohibit an accurate estimation of standard error using an LS estimator, and thus cause a poor power in the subsequent testing [61].

For this reason, developing robust regression estimators attracts a lot of attention in statistics communities; the accomplishments of prior research efforts are summarized by Maronna et al. [62]. In order to characterize the robustness of a regression method, Donoho and Huber introduced the concept of finite sample breakdown point, which can be considered as the smallest fraction of outliers that can ruin an estima-

tion (a mathematical definition is presented in Section IV.A) [39]. Intuitively, the higher the breakdown point value, the more robust an estimator is, and the more outliers an estimator can tolerate. The class of the so-called *high* breakdown point estimators includes the least trimmed squares (LTS) estimator and the least median squares (LMS) estimator, which adopt the robustness mechanism of utilizing a subset of measurements [32].

The breakdown point associated with a robust regression is not only determined by the type of estimator to be used, but also depends on the structure of design matrix $\mathbf{X}$. There have been a few papers on the robustness of a regression considering the structure of the design matrix [40, 41]. Denote by $\mathcal{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ the collection of known or observed information contained in $\mathbf{y}$ and $\mathbf{X}$. Mili and Coakley considered $\mathcal{Z}$ in *general position* if any p row vectors of $\mathbf{X}$ are linearly independent, and in *reduced position*, if some p row vectors in $\mathbf{X}$ are linearly dependent [41]. Furthermore, they called Model (1.1) a *structured* linear regression model when $\mathcal{Z}$ is in reduced position.

Structured linear regression arises in many applications including experimental designs. The author's own experiences with sensor network applications testify to the importance of structured linear regression. As we will discuss more in a latter section, a sensor network can often be modeled using a matrix representation, where $\mathbf{X}$ is not a data matrix of regressors, but a system matrix decided by the physical laws characterizing the relationship between the sensors and the physical quantities they are measuring. The linear model representing a sensor network is mathematically equivalent to the regression model in (1), but its counterpart of $\mathbf{X}$ is rarely in general position.

For this reason, developing robust regression estimators attracted attention. In order to quantify the robustness of a regression method, [39] introduced the

concept of a finite sample breakdown point, summarized as follows. Denote by $\mathcal{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ the collection of known or observed information contained in $\mathbf{y}$ and $\mathbf{X}$ and by $T(\mathcal{Z})$ a regression estimator using $\mathcal{Z}$. So $\hat{\boldsymbol{\beta}} = T(\mathcal{Z})$. When data points in $\mathcal{Z}$ are contaminated by outliers, the estimation of $\boldsymbol{\beta}$ may substantially deviate from its supposed true value. Suppose m data points of $\mathcal{Z}$ are contaminated, meaning that any m data points can be replaced by arbitrary values. We label the contaminated data set as $\mathcal{Z}'_m$. The resulting estimator is $T(\mathcal{Z}'_m)$. The maximum difference between $T(\mathcal{Z})$ and $T(\mathcal{Z}'_m)$ is denoted by bias$(m; T)$, defined as

$$\text{bias}(m; T) = \sup_{\mathcal{Z}'_m} ||T(\mathcal{Z}'_m) - T(\mathcal{Z})||, \tag{4.1}$$

where the supremum is over all possible $\mathcal{Z}'_m$ for a given m, and $\|\cdot\|$ means a Euclidean norm. The breakdown point of a regression estimator, denoted by $\epsilon^*_n(T, \mathcal{Z})$, is defined as

$$\epsilon^*_n(T, \mathcal{Z}) = \min\{\frac{m}{n} : \text{bias}(m; T) \text{ is infinite}\}. \tag{4.2}$$

Intuitively, the higher the breakdown point value, the more robust an estimator is, and the more outliers an estimator can tolerate.

Rousseeuw proved that the maximum attainable breakdown point, $\epsilon^*_{\max}$, is

$$\epsilon^*_{\max,n} = \frac{\lfloor (n - p + 2)/2 \rfloor}{n}, \tag{4.3}$$

where $\lfloor a \rfloor$ denotes the largest integer smaller than or equal to a [32]. He further developed the Least Trimmed Squares (LTS) estimator that could attain the maximum breakdown point. The LTS estimator is given by

$$\min \sum_{i=1}^{h} w^2_{(i)}, \tag{4.4}$$

where h is called a trimming parameter, $w^2_{(1)} \leq w^2_{(2)} \leq \ldots \leq w^2_{(n)}$ are the squared

residuals $(y_i - \mathbf{x}_i\hat{\boldsymbol{\beta}})^2$ for $i = 1, \ldots, n$, arranged in ascending order. To achieve the highest breakdown point, Rousseeuw stated that the trimming parameter should be so chosen that $h = [n/2] + [(p+1)/2]$ [32].

It turns out that the maximum breakdown point identified in [32] actually only works for data points $\mathcal{Z}$ in general position. Mili and Coakley studied the situation when $\mathcal{Z}$ is in reduced position [41] and revised the maximum breakdown point as

$$\epsilon^*_{\max,n} = \frac{\lfloor (n - L + 1)/2 \rfloor}{n}, \tag{4.5}$$

where L is the maximum number of row vectors in $\mathbf{X}$ that lie on a $(p-1)$-dimensional hyperplane passing through the origin. The $\epsilon^*_{\max,n}$'s in Equation (4.3) and (4.5) are the same when the linear model is in general position because $L = p - 1$ under that condition. Mili and Coakley further stated that an LTS estimator can attain the maximum breakdown point $\epsilon^*_{\max}$ if the LTS trimming parameter h in (4.4) is chosen so that $h_L \leq h \leq h_U$, where $h_L = [(n + L + 1)/2]$ and $h_U = [(n + L + 2)/2]$ [41]. Apparently, for strictured linear regressions, the role of l is critical in determining the breakdown point condition and devising a robust estimator. Improper calculation of L could lead to an improper value of h and cause a robust estimator lose its robustness.

B. Fault tolerance capability and redundancy degree

Robust estimation can easily find its application to sensor networks. In a large-size sensor network, the chances are high that some sensors may malfunction during its service life. Without isolating and eliminating sensor anomalies, malfunctioning sensors could mislead our understanding of the process status, and consequently, cause frequent false alarms and jeopardize productivity. A robust estimator is important and desirable in the presence of sensor anomalies to ensure a high-fidelity estimation

of the underlying process status.

Given the definition of the redundancy degree $\eta(\mathbf{X})$ in Chapter I, we can easily find the following relationship between L and $\eta(\mathbf{X})$.

Proposition 16 *When L and η are associated with the same design matrix $\mathbf{X}$,*

$$L = n - \eta - 1.$$

Proof. Since a hyperplane is a maximal non-spanning set, L is equal to the maximum cardinality of the hyperplanes of $M[\mathbf{X}^T]$. By Lemma 2, $L = n - g^*(M[\mathbf{X}^T])$. Therefore, using Proposition 1, we can conclude that $\eta(\mathbf{X}) = g^*(M[\mathbf{X}^T]) - 1 = n - L - 1$. □

The above relationship in Proposition 16 associates a physical interpretation with L; the quantity L is the complementary measure of the redundancy degree in measurements. When $\mathcal{Z}$ is in general position, the redundancy η is simply the difference between the number of sensor readings and the number of unknowns (i.e., $n-p$) so that $L = n - (n-p) - 1 = p - 1$. When $\mathcal{Z}$ is in reduced position, the linear dependence in $\mathbf{X}$ causes a reduction in the degree of redundancy in a system so that η decreases, or L increases.

We define the maximum achievable fault tolerance capability using a simple transformation of $\epsilon^*_{\max,n}$, that is, the maximum fault tolerance capability $\tau(\mathbf{X})_{\max}$, or simply $\tau_{\max}$, is defined as

$$\tau_{\max} \triangleq n \cdot \epsilon^*_{\max,n} - 1. \tag{4.6}$$

The difference between the breakdown point and $\tau_{\max}$ is trivial. We write the fault tolerance capability in an integer number, whereas a breakdown point is written in a fraction. In this way, practitioners easily interpret the fault tolerance capability - it is the maximum number of sensor failures, which the design of a sensor network can tolerate. The $\tau_{\max}$ is different from $n \cdot \epsilon^*_{\max,n}$ by one because one usually measures the

fault tolerance capability of a system right before it breaks down. Note that equation (4.6) identifies the fault tolerance capability of a sensor network configuration modeled using the design matrix $\mathbf{X}$. Translating equation (4.5) to the fault tolerance capability, it reads

$$\tau_{\max} = \lfloor \delta(\mathbf{X})/2 \rfloor. \tag{4.7}$$

Equation (4.7) indicates that the maximum number of sensor failures a system can tolerate is close to half of the redundancy degree of the system. This seems different from the message conveyed in some prior work, e.g., by Staroswiecki et al. (2004), where the degree of sensor redundancy is considered the same as the system capability of fault tolerance. When the redundancy degree and the fault tolerance capability are considered the same, underlying assumption is that we know which sensor fails and which sensor does not. However, if we do not know which sensor fails or which measurement is anomalous, but assume that they are equally likely to fail, then equation (4.7) should be used to determine the system's capability of fault tolerance.

In general, the fault tolerance capability depends on two factors: the sensor network configuration and the estimator T to be used; the notation $\tau(\mathbf{X}, T)$ represents both dependencies. Obviously, $\tau(\mathbf{X}, T)$ is generally less than $\tau_{\max} = [\delta(\mathbf{X})/2]$ but can reach the maximum level when a proper robust estimator is used. An LTS estimator can achieve $\tau_{\max}$ as long as its trimming parameter h is properly chosen. We translate the two bounds for h in [41] using the redundancy degree, and they are $h_L = [(2n - \delta(\mathbf{X}))/2]$ and $h_U = [(2n - \delta(\mathbf{X}) + 1)/2]$.

C. LTS estimation

Chapter III of this dissertation includes a computational procedure for the redundancy degree $\eta(\mathbf{X})$. This procedure is important in assessing the breakdown point, as well as determining the trimming parameter h for an LTS estimator. When one chooses h such that $h_L \leq h \leq h_U$, the resulting LTS estimator attains the maximum breakdown point $\epsilon^*_{\max,n}$ in equation (4.5). A question naturally follows; what happens if one uses an h outside the optimal range, namely either $h < h_L$ or $h > h_U$?

Since h is the number of data points used for estimation, one may expect a higher breakdown point when using a smaller h and a lower breakdown point when using a larger h. It is in fact true that using $h > h_U$ results in a lower breakdown point than $\epsilon^*_{\max,n}$, as formally stated in Theorem 5.2 in [41].

We know that using $h < h_L$ cannot guarantee the same level of robustness as using the optimal h. But one may argue that using $h < h_L$ could still be beneficial; the reasoning is as follows. When using $h < h_L$, LTS estimation may break down only if the outliers correspond to a set of "worst-case" rows of $\mathbf{X}$ but could still provide robustness to a larger number of outliers if they fall in other rows. If the chance for outliers to fall in the "worst-case" row is small (and one might expect so when $\eta << n-p$), then in practice setting a smaller h might be a better choice than using $h_L \leq h \leq h_U$

It becomes clear later that the above argument of using a small h (smaller than h_L) could be difficult to implement in practice. The difficulty lies in two aspects: (1) it is a computationally challenging task to identify all the "worst-case" rows where outliers can cause a robust regression to break down; (2) the chance for outliers to fall in the "worst-case" row could be large. Thus, we feel it is a safer choice, and thus a sound practice, to use $h_L \leq h \leq h_U$ unless the "worst-case" rows can be easily

enumerated and the chance of an estimator breaking down can be easily assessed. We would like to elaborate ourselves using the following example.

Suppose we have a regression model, of which the design matrix is

$$\mathbf{X}^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{4.8}$$

Then, $n = 10, p = 4$, and $\eta(\mathbf{X}) = 2$. Note that here η is smaller than half the value of $n - p$. For this model, $h_U = h_L = 9$ and $\epsilon^*_{\max,n} = 2/n$ (i.e., $\tau_{\max} = 1$); this implies that LTS estimation with $h = 9$ tolerates one outlier and may break down with two outliers. Now the question is what happens if we use $h = 8$, a value smaller than h_L. It turns out that the LTS estimator with $h = 8$ may not tolerate even a single outlier. To see this, suppose y_3, the third element in $\mathbf{y}$, becomes an outlier (i.e., the value of y_3 can take any value). Then, it is easy to see that, for an arbitrary value of y_3, trimming off y_5 and y_6 and setting $\hat{\beta}_2 = y_3$ will always lead to an exact fit of y_3 (i.e., $\mathbf{x}_3\hat{\boldsymbol{\beta}} - y_3 = 0$), while producing good fits of the other observations (i.e., their residuals are finite). Thus, the LTS estimation with $h = 8$ is not robust against one outlier if the outlier occurs at y_3. Following the same spirit, one outlier occurring at y_1, y_2, y_5, or y_6 can cause the LTS estimation with $h = 8$ to break down. In comparison, when $h = 9$ and y_3 is an outlier, at most one of y_5 and y_6 can be discarded. Suppose that y_5 is trimmed off. Although setting $\hat{\beta}_2 = y_3$ can still lead to an exact fit of y_3, it will also result in a large residual at y_6. Then, the best fit will tend to discard y_3, which is the outlier. This is why $h = 9$ provides robustness to single outliers.

In the above example, the"worst-case" rows are $\{1, 2, 3, 5, 6\}$; finding the "worst-case" rows using hyperplanes of $M[\mathbf{X}^T]$ are explained later in this section. In other

words, when one or two outliers occur at the other five rows, namely $\{4, 7, 8, 9, 10\}$, the LTS estimator will still be able to produce a robust estimation using $h = 8$. This illustrates the argument at the beginning of this section that using a smaller h can sometimes provide a higher degree of robustness. But the chance for an LTS to still produce a robust result could be low. For the example at hand, when there is indeed only one outlier, the chance for the outlier to occur in one row of $\{1, 2, 3, 5, 6\}$ is 50% assuming that the outlier has an equal likelihood to fall in any one of the ten rows. When there are two outliers, the chance that none of the outliers falls in the "worst-case" rows is $\frac{C_5^2}{C_{10}^2} = 22\%$, meaning that the chance for an LTS estimator to break down is at least $1 - 22\% = 78\%$. This example illustrates that even a slightly smaller h may cause the LTS algorithm to lose its robustness with a significant chance. If a much smaller h is used, the problem will become severer.

In order to assess the likelihood that the resulting LTS estimation breaks down, one needs to enumerate all the "worst-case" rows. Doing so in turn requires to enumerate all the hyperplanes of $M[\mathbf{X}^T]$. Unfortunately, finding all hyperplanes is generally computationally challenging (more difficult than finding the redundancy degree) and typically needs to involve theories in combinatorial mathematics that practitioners may not be familiar with. For this reason, using the optimal h (i.e., $h_L \leq h \leq h_U$) is a safe and sound choice. For the interested readers who would like to know how to enumerate all hyperplanes, please refer to [60] or refer to [63] for a cocircuit enumeration algorithm because a cocircuit is a dual form of a hyperplane.

D. LTS estimation using a lower bound

Suppose we do not know the exact value of the redundancy degree but the lower bound of the redundancy degree is known. Denote by $\nu(\mathbf{X})$ a lower bound of the

redundancy degree we obtained; i.e., $\nu(\mathbf{X}) \leq \eta(\mathbf{X})$. We may use $\nu(\mathbf{X})$ for the place of the redundancy degree when choosing h such that

$$h'_L \leq h \leq h'_U, \tag{4.9}$$

where $h'_L = \lfloor (2n - \nu(\mathbf{X}))/2 \rfloor$ and $h'_U = \lfloor (2n - \nu(\mathbf{X}) + 1)/2 \rfloor$. Then, choosing $h \geq h'_L$ does not result in unstable result as we have seen in Section IV.C since $h'_L \geq h_L$. However, this h value possible greater than h_U, and the maximum fault tolerance capability may not be attained.

If we choose h from $[h'_L, h'_U]$, it is likely that the chosen h is greater than or equal to h_U. When $h \geq h_U$, the breakdown point of an LTS estimator becomes [41]

$$\epsilon^*(T_{LTS(h)}, \mathcal{Z}) = \frac{n - h + 1}{n}, \tag{4.10}$$

where $T_{LTS(h)}$ is an LTS estimator whose trimming parameter is chosen from the new range. Note that the larger the h, the worse off the breakdown point. In other words, the consequence of using the lower bound is that the resulting LTS estimator will reach a robustness level lower than that of the optimally tuned LTS estimator.

Under the circumstances when h_L is not known (because $\eta(\mathbf{X})$ is unknown) but only h'_L and h'_U are known, the safest choice is to let $h = h'_L$. We denote this value as h^*; the maximum breakdown point of the LTS estimator with $h = h^*$ depends on where h'_L lies.

When $\nu(\mathbf{X})$ is an even number, h'_L=$h'_U \geq h_U$ so that equation (4.10) is valid. Then, the maximum breakdown point of the LTS estimator is

$$\epsilon^*(T_{LTS(h^*)}, \mathcal{Z}) = \frac{[(\nu(\mathbf{X}))/2] + 1}{n}. \tag{4.11}$$

From (4.6) and (4.10), the fault tolerance capability using h^* is

$$\tau(T_{LTS(h^*)}, \mathbf{X}) = [\nu(\mathbf{X})/2] \tag{4.12}$$

When $\nu(\mathbf{X})$ is an odd number, h'_U=h'_L+1 and the situation becomes complicated. If h'_L is still greater than or equal to h_U, the maximum breakdown point of the LTS estimator becomes

$$\epsilon^*(T_{LTS(h^*)}, \mathcal{Z}) = \frac{[(\nu(\mathbf{X}) + 1)/2] + 1}{n}, \tag{4.13}$$

and the corresponding fault tolerance capability is

$$\tau(T_{LTS(h^*)}, \mathbf{X}) = [(\nu(\mathbf{X}) + 1)/2]. \tag{4.14}$$

If h'_L is less than h_U, it implies that $h'_L = h_L$. This is because h_L and h_U are both integers that are apart by at most one, and so are h'_L and h'_U. The situation that $h'_L = h_L$ could happen only when the lower bound $\nu(\mathbf{X})$ is the same as $\eta(\mathbf{X})$. So the maximum breakdown point of the LTS estimator and the fault tolerance capability are the same as those in (4.5) and (4.6), respectively.

The difficulty associated with the case where $\nu(\mathbf{X})$ is an odd number is that one does not know the relationship between h'_L and h_U since $\eta(\mathbf{X})$ is unknown, which is the reason why $\nu(\mathbf{X})$ is calculated in the first place. If we always use equations (4.13) and (4.14), it will give an overestimated breakdown point or fault tolerance capability when $h'_L < h_U$, or equivalently, when $\nu(\mathbf{X}) = \eta(\mathbf{X})$. This happens because (4.14) is derived from (4.10), which is only valid for an $h \geq h_U$ (and this condition is difficult to verify). Given this difficulty, we suggest using equations (4.11) and (4.12) to calculate the breakdown point and the fault tolerance capability at all times but acknowledge that the actual LTS estimator can probably perform slightly better than what the calculated value suggests.

E. Summary

This chapter introduces robust measures - breakdown point and fault tolerance capability - and LTS estimation that attains the maximum breakdown point or fault tolerance capability. This chapter shows that the breakdown point of a LTS estimator depends on the redundancy degree, and thus, the trimming parameter of LTS estimation should be determined based on the redundancy degree that is $h_L \leq h \leq h_U$. Should one use $h < h_L$, the resulting LTS estimation loses its robustness, and the possibility of the breakdown can be high as stated in Section IV.C.

Section IV.D presents LTS estimation using a lower bound of the redundancy degree. A lower bound of the redundancy degree does not provided the maximum level of the robustness but provides a certain level of robustness. However, if we use an approximated value of a redundancy degree, the resulting LTS estimation may not be robust.

CHAPTER V

APPLICATIONS

This chapter presents two applications of robustness analysis on clustered sensor networks. Section II.A provides a coordinate sensor network on a multi-station assembly process for manufacturing quality control. Section II.B provides two wireless sensor networks for a calibration problem in localization. All these case studies show that the research work, in this dissertation, brings robustness to sensor network applications and demonstrates how the decomposition algorithms benefit us when computing the redundancy degree.

A. Multi-station assembly process

In order to make a connection to engineering applications, we choose to illustrate our development using a multi-station assembly process equipped with a distributed, redundant sensor system, as shown in Figure 7. This assembly process consists of three stations: on Station 1, two parts are assembled and the resulting subassembly is transferred to Station 2; on Station 2, the subassembly is assembled with two more parts; on Station 3, there is no assembly operation but the final assembly is positioned for quality inspection. Coordinate sensors are placed not only on Station 3, but also on Stations 1 and 2. A coordinate sensor measures a dimensional deviation of the part either in the x-direction or in the z-direction. There are a total of n=26 sensors; their positions and measurement directions are indicated by an arrow in Figure 7.

For this example, the unknown parameter $\boldsymbol{\beta}$ denotes the deviations associated with the fixture locators that hold the parts during the assembly operation. Each part or each finished subassembly is positioned by a pair of locators, consisting of a 4-way locator that controls the part motions in both x- and z-directions, and a 2-way

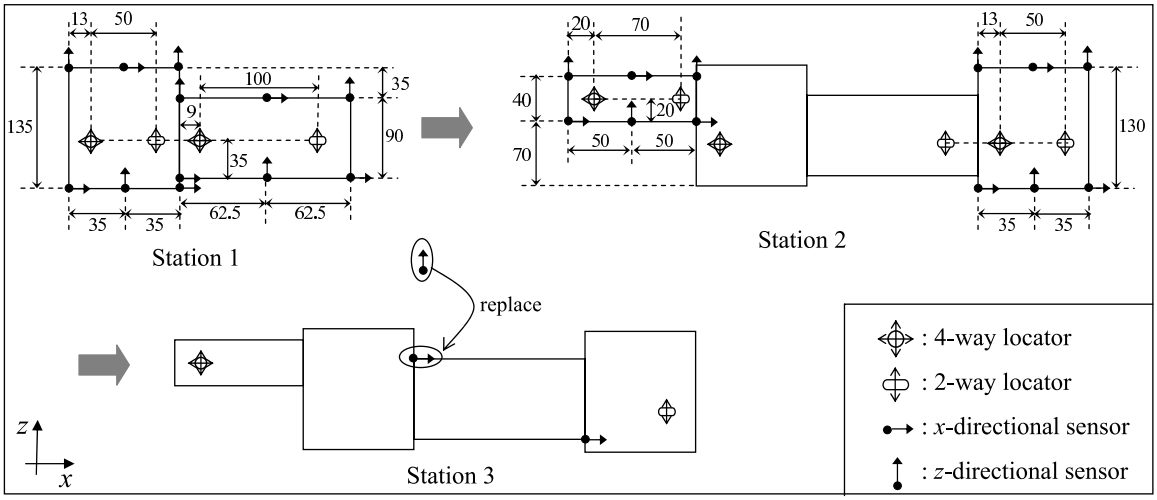


Fig. 7. A three station assembly process

locator that controls the part motion in the z-direction alone. The potential locator deviations to be estimated are represented by a double-arrowed bar on a circle or on a slot that indicates the location of a fixture locator. There are a total of $p = 12$ potential deviations on the three stations, and the dimension of $\boldsymbol{\beta}$ is 12×1. The design matrix $\mathbf{X}$ for this system is shown in Figure 8. The detailed modeling procedure to obtain this $\mathbf{X}$ matrix can be found in [64].

1. Calculating redundancy degree

First, use the bigraph-based method in Chapter II to transform $\mathbf{X}$ into a BBF. For the $\mathbf{X}$ in Figure 8, $R = \text{Row}(\mathbf{X}) = \{1, 2, \ldots, 26\}$ and $C = \text{Col}(\mathbf{X}\} = \{1, 2, \ldots, 12\}$. Using the bipartite graph procedure in Section 2.1, we find $S = \{10, 14\}$. Then, remove S and decompose the reduced bipartite graph associated with $\mathbf{X}[R - S, C]$. Finally, we obtain $C_1 = \{1, 2, 3\}$, $C_2 = \{4, 5, 6\}$, $C_3 = \{7, 8, 9\}$ and $C_4 = \{10, 11, 12\}$. Figure 9 shows the design matrix in a bordered block form. Hence, $k = 4$ and

$$\mathbf{X} = \begin{pmatrix}
1 & 1.8 & -1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1.8 & -1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & .2857 & -.2857 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & .2857 & -.2857 & 0 & 0 & 0 \\
1 & -.9 & .9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.26 & -.26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & .56 & .44 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & .55 & -.55 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -.35 & .35 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -.3313 & 0 & 1 & .55 & -.2186 & -1.0041 & -.0297 & 0 & .004 & 0 & .0297 \\
0 & 0 & 0 & 0 & 1.09 & -.09 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & .5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -.375 & 1.375 \\
-1 & .2108 & 0 & 1 & -.35 & .1392 & -.9676 & .238 & 0 & -.0324 & 0 & -.238 \\
0 & 0 & 0 & 0 & -.16 & 1.16 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -.2857 & .2857 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2857 & -.2857 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .5714 & .4286 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -.1429 & 1.1429 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2.25 & -2.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & -2 \\
0 & 0 & 0 & 0 & .465 & .535 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -.14 & 1.14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & .55 & -.55 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -.1.125 & 1.125 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.375 & -.375
\end{pmatrix}$$

Fig. 8. The design matrix of the multistation assembly process

$$
\mathbf{X} = \left(\begin{array}{ccc|ccc|ccc|ccc}
1 & 1.8 & -1.8 & & & & & & & & & \\
1 & 1.8 & -1.8 & & & & & & & & & \\
1 & -.9 & .9 & & & & & & & & & \\
0 & 1.26 & -.26 & & & & & & & & & \\
0 & .56 & .44 & & & & & & & & & \\
0 & -.14 & 1.14 & & & & & & & & & \\
\hline
 & & & 1 & .55 & -.55 & & & & & & \\
 & & & 1 & .55 & -.55 & & & & & & \\
 & & & 1 & -.35 & .1392 & & & & & & \\
 & & & 0 & 1.09 & -.09 & & & & & & \\
 & & & 0 & .465 & .535 & & & & & & \\
 & & & 0 & -.16 & 1.16 & & & & & & \\
\hline
 & & & & & & 1 & .2857 & -.2857 & & & \\
 & & & & & & 1 & .2857 & -.2857 & & & \\
 & & & & & & 1 & -.2857 & .2857 & & & \\
 & & & & & & 0 & 1.2857 & -.2857 & & & \\
 & & & & & & 0 & .5714 & .4286 & & & \\
 & & & & & & 0 & -.1429 & 1.1429 & & & \\
\hline
 & & & & & & & & & 1 & 2.25 & -2.25 \\
 & & & & & & & & & 1 & 2 & -2 \\
 & & & & & & & & & 1 & -1.125 & 1.125 \\
 & & & & & & & & & 0 & 1.375 & -.375 \\
 & & & & & & & & & 0 & .5 & .5 \\
 & & & & & & & & & 0 & -.375 & 1.375 \\
-1 & -.3313 & 0 & 1 & .55 & -.2186 & -1.0041 & -.0297 & 0 & .004 & 0 & .0297 \\
-1 & .2108 & 0 & 1 & -.35 & .1392 & -.9676 & .238 & 0 & -.0324 & 0 & -.238
\end{array}\right)
$$

Fig. 9. The design matrix in a bordered block form

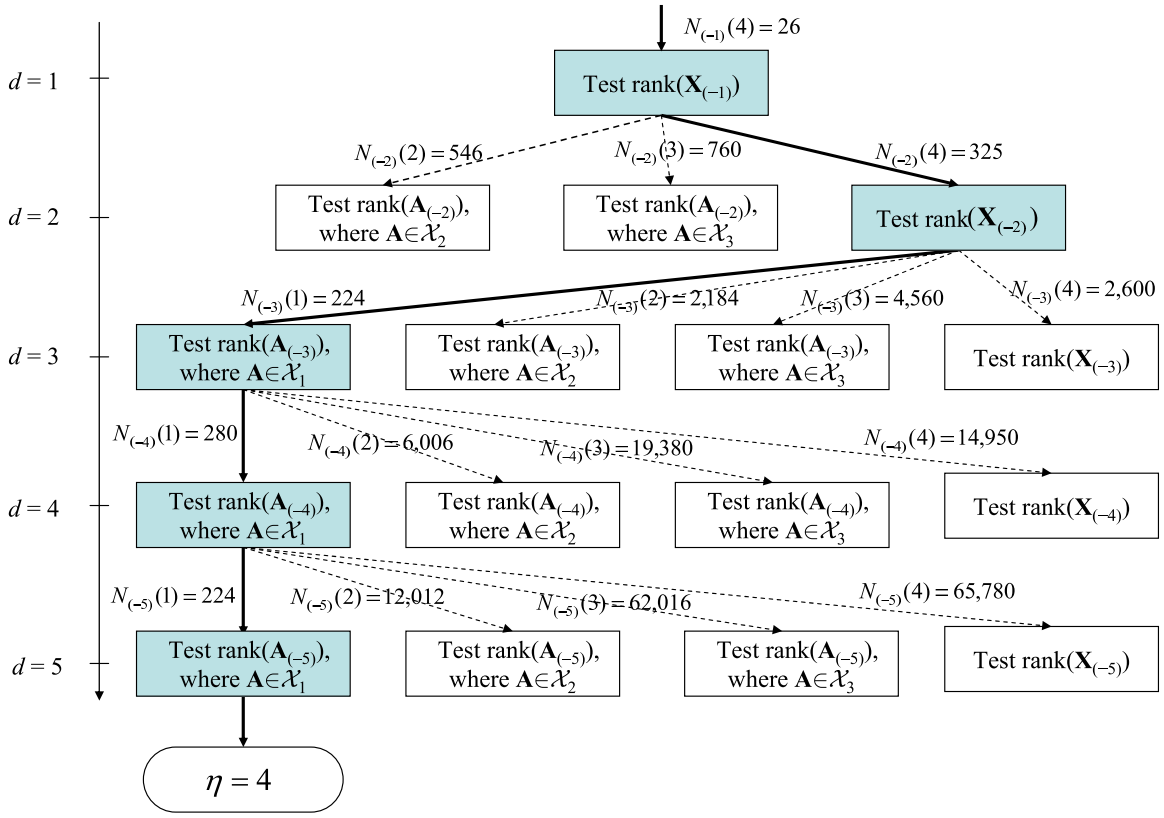


Fig. 10. The decomposition algorithm on multi-station assembly

$\frac{k}{k-1}|S| \approx 2.67$. Second, use Algorithm 3 to find $\eta(\mathbf{X})$; the computation procedure is illustrated in Figure 10. Given $\frac{k}{k-1}|S| \approx 2.67$, it means that the bound on d, allowing us to switch submatrices testing, is $\lceil \frac{k}{k-1}|S| \rceil - 1 = 2$. Thus, we run the rank testing on $\mathbf{X}$ for $d = 1$, and since $\mathbf{X}_{(-1)}$ is of full column rank, we conclude that $\eta(\mathbf{X}) > 0$. Starting with $d = 2$, we can test q-block submatrices. When $d = 2$, we may test submatrices in $\mathcal{X}_2$, $\mathcal{X}_3$, or $\mathcal{X}_4$ and it turns out that $N_{(-2)}(4)$ is the smallest one among the three alternatives; from Figure 10, it reads that $N_{(-2)}(2) = 6 \times C_{14}^2 = 546$, $N_{(-2)}(3) = 4 \times C_{20}^2 = 760$, and $N_{(-2)}(4) = C_{26}^2 = 325$. Hence, $q^* = 4$ for $d = 2$ and the ranks of $\mathbf{X}_{(-2)}$ are tested. Again, we find that $\eta > 1$. The procedure is repeated for

Table III. Computation time for $\eta(\mathbf{X})$ by exhaustive search and bound-&-decompose

Design matrix				Computation time	
Dimension	$\|S\|$	k	η	exhaustive search	bound & decompose
26×12	2	4	4	8 sec.	.1 sec.
67×24	3	8	7	122.5 hours	6.1 min.
222×55	2	11	14	> 120 hours	16.5 min.
1009×252	1	41	15	> 120 hours	38.2 min.

$d = 3, 4$, and 5. In those cases, $q^* = 1$. Eventually, we test the ranks of submatrices in $\mathcal{X}_1$ for $d = 5$, and find that $\eta = 4$.

The computation path is highlighted by the solid arrows in Figure 10, where the dashed line indicates other possible alternatives. Adding the numbers on the solid arrows gives us the the number of the maximum iterations used by the algorithm, which is 1,079. By comparison, had we used the exhaustive rank testing on $\mathbf{X}$ entirely, the iteration number should have been at most 83,681, which actually equals the addition of the numbers on the rightmost arrows. The algorithm coded in MATLAB goes through 904 actual iterations in 1 seconds, while the exhaustive rank testing takes 8 seconds to finish 45,565 actual iterations. Should the size of $\mathbf{X}$ be larger, the potential saving could be more substantial. We also create larger systems, of which the design matrices are 67×24, 222×24, and 1009×252. The computation results are summarized in Table III.

2. LTS estimation

In order to utilize the LTS estimator in its maximum capacity, we need to choose the parameter h to be in the range $h_L = \lfloor (2n - \eta(\mathbf{X}))/2 \rfloor$ and $h_U = \lfloor (2n - \eta(\mathbf{X}) + 1)/2 \rfloor$.

Table IV. MSE of LS estimation, LTS($h = 19$) estimation, and LTS($h = 24$) estimation

Number of measurement outliers	LS	LTS($h = 19$)	LTS($h = 24$)
0	.0020	.0074	.0039
1	.0370	.0407	.0116
2	.0771	.0626	.0127

Given $\delta = 4$, we have $h = 24$. It suggests that the LTS with $h = 24$ can tolerate up to $n \cdot \epsilon_{\max,n} = 2$ measurement outliers. However, had we inappropriately chosen the h parameter using the formula in [32], which is designed for general position, then $h = [n/2] + [p/2] = 19$.

Using these two LTS estimators and the ordinary least squares estimator, we run a simulation study to observe their robustness. We simulated $N = 1,000$ observations; the sensor noise $\mathbf{e}$ under a normal work condition is assumed to be normally distributed with a zero mean and a small standard deviation of .02, and measurement outliers are simulated by adding a substantial deviation of magnitude .3 to some of the measurements. Since the LTS algorithm can tolerate up to two outliers, we simulate the cases with no outlier, one outlier, and two outliers, respectively. To compare the performance of these estimators, the mean of squared errors (MSE) is used where

$$\text{MSE} = \frac{1}{N}\sum_{t=1}^{N}(\boldsymbol{\beta}_t - \hat{\boldsymbol{\beta}}_t)^T(\boldsymbol{\beta}_t - \hat{\boldsymbol{\beta}}_t).$$

Table IV shows the performances of the LS estimator, the LTS estimator with $h = 19$, and the LTS estimator with $h = 24$. With the presence of measurement outliers, the LTS estimator ($h = 24$) is more robust than both the LS estimator and the LTS estimator ($h = 19$), as indicated by its relatively flat MSE value. The MSE's

of the other two estimators escalate rapidly when there exists an outlier. Without an outlier, an LS estimator generally outperforms an LTS estimator because the latter utilizes only a subset of sensor measurements so that its efficiency suffers a little bit. Using the LTS estimator ($h = 19$) is even worse off than the LS estimator because MSE values of the LTS estimator ($h = 19$) and the LS estimator are similar; however, when outliers are absent, the LTS ($h = 19$) estimator's MSE is four times larger than that of the LS and two times larger than that of LTS ($h = 24$). This provides an example that a robust estimator could lose its robustness if the structure in the design matrix is not taken into account.

As to the question if one could use an $h < 24$ to tolerate more outliers, the answer rests with the assessment of how likely an LTS estimator breaks down. However, enumerating the hyperplane sets is not so easy to perform for a matrix with 26 rows. Thus, again $h = 24$ is a safer choice.

B. Robust calibration for localization of clustered wireless sensor network

Wireless technologies have added flexibility to the design and operation of sensor networks. Equipped with micro-electro-mechanical systems (MEMS), a wireless sensor node becomes small, mobile, and multi-functional. One of the most significant changes caused by the wireless technologies is the implementation of *ad-hoc* networking, which refers to those having a network topology that can change frequently [22]. The frequent changes in the topology of an ad-hoc network naturally call for a solution to the *localization* or *location tracking* problem, because knowing the positions of individual sensors is often the pre-requisite to many subsequent decision makings. Installing a global positioning system (GPS) [65] could be a solution, but the heavy power consumption and high equipment cost associated with a GPS deem it imprac-

tical to install on every micro-sensor node. In practice, GPS receivers may be used only on a small portion of sensor nodes, known as *anchor nodes*, in a network [66]. The location of a non-anchor node can be decided and tracked relative to the anchor nodes; first, measure the distances between itself and several anchor nodes; then, compute its location based on certain geometry principle (e.g., hyperbolic trilateration, triangulation, and multilateration).

One method of measuring the between-node distance is to use two types of signals, a radio frequency (RF) and an acoustic signal, which travel at different speeds. The time difference of arrival (TDOA) between the two signals is then used to calculate the between-node distance [67]. One problem associated with this distance measuring approach is its inaccuracy. For example, RF signals are attenuated by metal objects [68] and the speed of acoustic signals are highly influenced by temperature and moisture [67]. The experiments performed by Whitehouse and Culler [69] showed that the error of a between-node distance measurement using acoustic time of flight could be as large as 300% of the true distance. To tackle this issue, Whitehouse and Culler [69] recommended using the following *calibration* procedure. In an off-line setting, the true distance between sensor nodes can be measured by independent and accurate means; then, a mathematical model mapping the distance to the true distance is established. The mapping model adjusts the distance measurements, during the service of sensor nodes, to a more accurate estimation of the true distances.

1. Calibration model

Denote by $d_{u,v}$ the measured distance between transmitter u and receiver v, by $y_{u,v}$ the true distance, and by e the random noise. Whitehouse and Culler [69] proposed

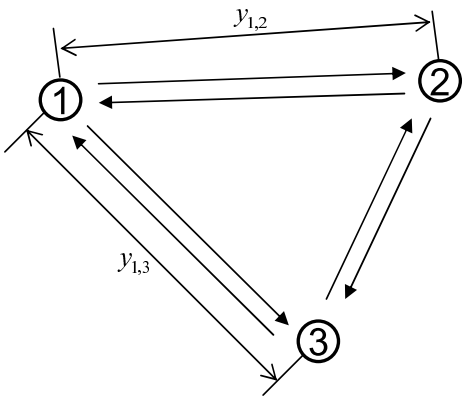


Fig. 11. Wireless sensor network (three sensors)

a linear calibration model such as

$$y_{u,v} = \alpha_u + \beta_v + \gamma_u d_{u,v} + \delta_v d_{u,v} + e, \tag{5.1}$$

where α_u and β_v are the bias of a transmitter u and a receiver v, and γ_u and δ_v are the gain of u and v, respectively.

In model (5.1), we set four parameters $(\alpha, \beta, \gamma, \delta)$ for a single wireless sensor node because a sensor node is assumed to work as both a transmitter and a receiver. Denote by Λ the set of the available true distances. In Figure 11, for example, suppose that the available true distances are $y_{1,2}$ and $y_{1,3}$; then, $\Lambda = \{y_{1,2}, y_{1,3}\}$. The actual number of the true distances used in the model doubles the cardinality of Λ since the between-node communications are two-way. We need to duplicate the elements in Λ such that $y_{u,v} = y_{v,u}$ and include them in a new vector $\boldsymbol{\lambda}$. For the example in Figure 11,

$$\boldsymbol{\lambda} = \begin{pmatrix} y_{1,2} & y_{2,1} & y_{1,3} & y_{3,1} \end{pmatrix}^T.$$

Denote by $\mathbf{b}$ the $4n \times 1$ vector of all the calibration parameters.

$$\mathbf{b} = \begin{pmatrix} \alpha_1 & \dots & \alpha_n & \beta_1 & \dots & \beta_n & \gamma_1 & \dots & \gamma_n & \delta_1 & \dots & \delta_n \end{pmatrix}^T.$$

In Figure 11, we have three sensor nodes, so $\mathbf{b}$ is a 12×1 vector. For a given $\boldsymbol{\lambda}$ and $\mathbf{b}$, it is straightforward to obtain the following matrix expression from (5.1);

$$\boldsymbol{\lambda} = \mathbf{G}\mathbf{b} + \mathbf{e}, \tag{5.2}$$

where $\mathbf{G}$ is a $n \times 4m$ matrix function and m is the number of sensors. For the sensor network in Figure 11, the $\mathbf{G}$ matrix is

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & d_{1,2} & 0 & 0 & 0 & d_{1,2} & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & d_{2,1} & 0 & d_{2,1} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & d_{1,3} & 0 & 0 & 0 & 0 & d_{1,3} \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & d_{3,1} & d_{3,1} & 0 & 0 \end{pmatrix}. \tag{5.3}$$

In order to uniquely estimate $\mathbf{b}$ in (5.2), $\mathbf{G}$ should be of full column rank. A necessary condition is to have $n \geq 4m$. This condition can be achieved if we include more sensors in a network. The number of pairwise distances, $\frac{m(m-1)}{2}$, increases faster than $4m$, the number of parameters.

In fact, even if $n \geq 4m$ holds, we may still run into a $\mathbf{G}$ that is not of full column rank. It turns out that the following linear dependence relationship exists;

$$\mathbf{g}_1 + \mathbf{g}_2 + \ldots + \mathbf{g}_m = \mathbf{g}_{m+1} + \mathbf{g}_{m+2} + \mathbf{g}_{2m} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T,$$

where $\mathbf{g}_i$ denotes the ith column vector of $\mathbf{G}$. This means that the original calibration formulation introduced in the wireless network literature ([69, 70]) over-parameterizes the system. In order to uniquely estimate the calibration parameters, additional constraints should be used to make the linear matrix of full column rank.

The constraint we use here comes from the small number of anchor nodes. According to [69], the anchor nodes can be *micro-calibrated*, meaning that these sensor nodes are regularly maintained, and the distance measurements between these maintained nodes accurately estimate the true distances. Then, the calibration parameters associated with the anchor nodes can be set as constants. In this research, the parameters of the micro-calibrated nodes are set as $(0, 0, .5, .5)$, which were suggested in [69]. Through hardware adjustment, one could possibly set the parameters to other constant values.

Let $\boldsymbol{\theta}$ be the vector of the calibration parameters of micro-calibrated sensor nodes and $\boldsymbol{\beta}$ be the vector of the unknown calibration parameters. Permute the rows in $\mathbf{b}$ such that

$$\mathbf{b}^T = \begin{pmatrix} \boldsymbol{\theta} & \boldsymbol{\beta} \end{pmatrix}^T.$$

Likewise, denote by $\mathbf{X}$ the submatrix of $\mathbf{G}$ associated with $\boldsymbol{\beta}$ and by $\mathbf{A}$ the submatrix of $\mathbf{G}$ associated with $\boldsymbol{\theta}$. Permute the columns in $\mathbf{G}$ such that

$$\mathbf{G} = \begin{pmatrix} \mathbf{A} & \mathbf{X} \end{pmatrix}.$$

Then, equation (5.2) can be rewritten as

$$\boldsymbol{\lambda} - \mathbf{A}\boldsymbol{\theta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}. \tag{5.4}$$

By defining $\mathbf{y} = \boldsymbol{\lambda} - \mathbf{A}\boldsymbol{\theta}$, we construct a model in (1.1), of which the number of elements in $\boldsymbol{\beta}$ is denoted by p $(< 4m)$.

2. Sub-calibration model in computing LTS estimators

The LTS estimator, as defined in (4.4), can be computed using a FAST-LTS routine [71] in several statistical software packages including R[1], S–Plus[2], and SAS[3]. In order to use the FAST-LTS routine, the user needs to input the trimming parameter h, which ensures the resulting estimation retains an appropriate level of robustness. Our analysis in the previous sections provides the proper way of determining the h used.

Once h is determined, computing an LTS estimator using the original $\mathbf{X}$ could still be computationally demanding because the algorithm needs to solve for the minimum sum of h squared residuals out of all the possible combinations. For a large matrix $\mathbf{X}$, we propose a sub-calibration model, which makes a few approximations but can greatly speed up the computation.

The sub-calibration applies an LTS estimator to the cluster matrix $\mathbf{X}^{(i)}$, instead of the original matrix $\mathbf{X}$, to estimate the parameters associated with sensor nodes in that particular cluster. In this approach, the calibration parameters are estimated separately for each cluster in the network. In so doing, we assume that the sensor nodes that are not in the ith cluster, but communicate with the sensor nodes in the ith cluster are micro-calibrated. This assumption could possibly reduce the accuracy of the estimation, but the nice aspect of using a robust estimator is that the estimation result is not very sensitive to this assumption. Since the LTS estimator eliminates suspicious data points, inaccurate values resulting from the assumption will be disregarded. As one will see in Section 4, doing the calibration for each cluster

[1]http://www.r-project.org

[2]http://www.splus.com

[3]http://www.sas.com

separately causes little difference as compared to calibrating the whole network using the FAST-LTS routine.

More specifics of the sub-calibration model is given as follows. Denote by $\boldsymbol{\beta}^{(i)}$ the parameters associated with the sensor nodes in the ith cluster, by $\mathbf{y}^{(i)}$ the true distances associated with the row labels R_i of $\mathbf{X}^{(i)}$, and by $\mathbf{e}^{(i)}$ the corresponding random noises. Because we assume that the sensor nodes connecting to the ith cluster are micro-calibrated, we deem that the parameters associated with those are constant. Then, we can write the following sub-calibration model for the ith cluster:

$$\mathbf{y}^{(i)} - \mathbf{X}[R_i, C - C_i] \begin{pmatrix} * & 0 & \dots & 0 & .5 & \dots & .5 & * \end{pmatrix}^T = \mathbf{X}^{(i)}\boldsymbol{\beta}^{(i)} + \mathbf{e}^{(i)}. \qquad (5.5)$$

In the above model, $(\ * \ \ 0 \ \ \dots \ \ 0 \ \ .5 \ \ \dots \ \ .5 \ \ * \ )^T$ has the same dimension with the sub-vector of $\boldsymbol{\beta}$ after $\boldsymbol{\beta}^{(i)}$ is excluded, in which the constants 0's and .5's are the parameters associated with the sensor nodes communicating with sensor nodes in the ith cluster, and the '$*$', meaning that an arbitrary value can be chosen, corresponds to the parameters of the nodes that are neither in the cluster nor communicating with the ith cluster. The computation benefit of using the sub-calibration model will become apparent in the following numerical examples.

3. Examples

Figure 12 shows the configuration of a wireless sensor network, where '$*$' denotes the location of a sensor node. Note that the coordinates in Figure 12 have been normalized for convenience, and thus the unit has no physical meaning. There are a total of 20 sensors, among which the sensors at locations $(1.0, 1.0)$ and $(2.6, 2.6)$ are micro-calibrated so that their calibration parameters are set to be constants, and the remaining 18 sensor nodes are ordinary ones whose parameters are to be estimated.

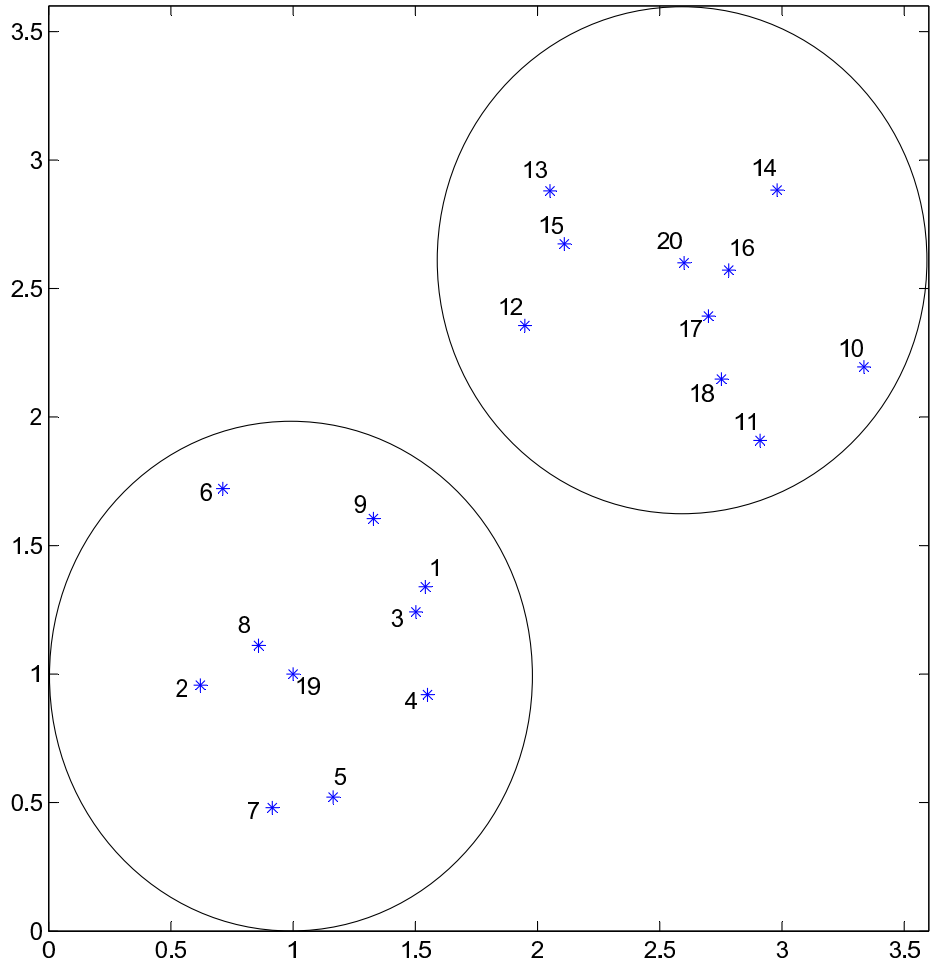


Fig. 12. Wireless sensor network (20 sensors)

The sensors are assigned an index from 1 to 20, where the numbers of 19 and 20 are reserved for the two micro-calibrated sensors.

The limit of the communication distance between a pair of sensor nodes is 1 (that is what we chose as the basic unit to scale the sensor network), meaning that the sensor nodes farther apart than this limit cannot communicate with each other. Applying this rule to the sensor network in Figure 12, we obtain a graph representation of the network in Figure 13; here, one can easily observe two clusters, where the between-

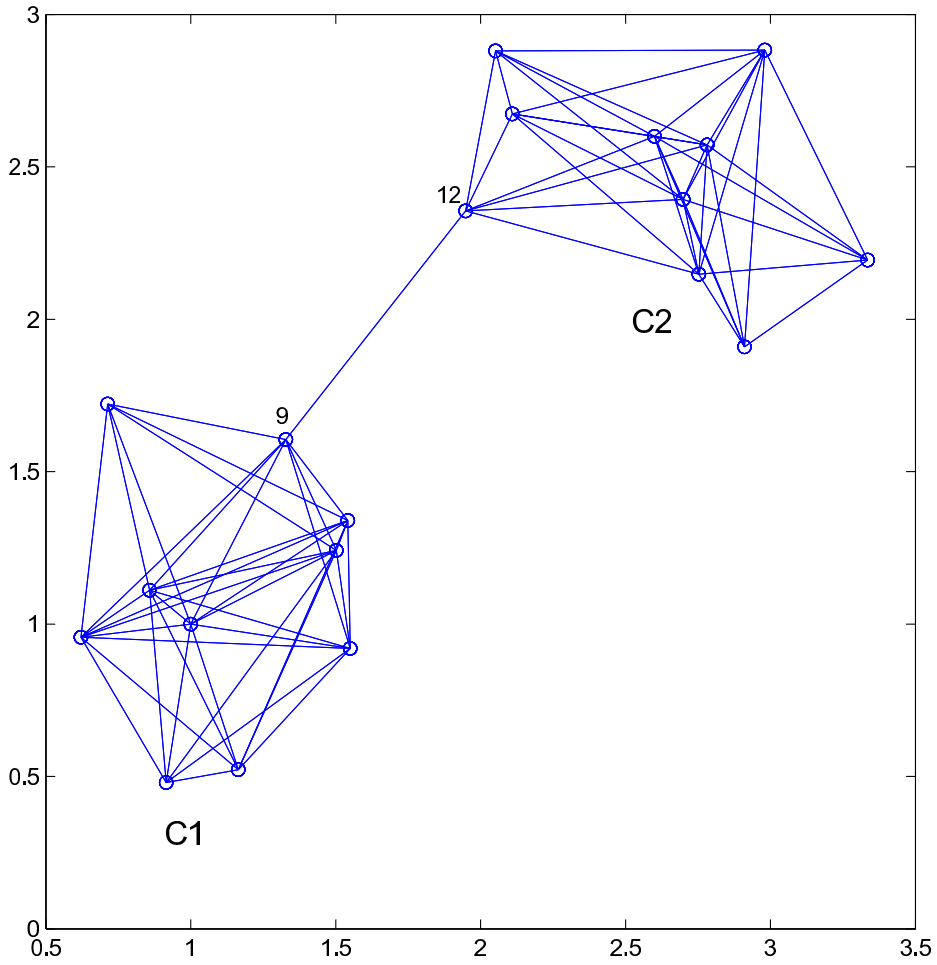


Fig. 13. Graph representation of the wireless sensor network in Figure 12

cluster communication is through the pair $\{9, 12\}$.

Given that there are four parameters associated with each sensor node to be calibrated, the dimension of $\boldsymbol{\theta}$ is $p = 72$. From Figure 13, we count $m = 72$ edges in the graph so that the size of $\mathbf{y}$ is $2m = 154$. This means the calibration matrix $\mathbf{X}$ is a 154×72 matrix. We choose to omit $\mathbf{X}$ here to save space.

Finding the redundancy degree of a 154×72 matrix is computationally difficult. The enumerative rank testing method will run up to $\sum_{d=1}^{\eta(\mathbf{X})+1} \binom{154}{d}$ rank testings. Since

we know $\eta(\mathbf{X}) = 4$ in this case from our latter analysis, $\sum_{d=1}^{5} \binom{154}{d} \simeq 7 \times 10^9$. Even for the best case scenario, where the enumerative algorithm finds $\eta(\mathbf{X})$ at its first rank test right after d reaches 5, it still needs to go through $\sum_{d=1}^{4} \binom{154}{d} + 1 \simeq 2.3 \times 10^8$ rank testings. The program is terminated after a day of computing since it has already taken far more time than the decomposition algorithm.

In order to use the algorithms presented in this paper, we need to identify the bordered block form of $\mathbf{X}$ first. In fact, because of the simple network configuration here, one can actually tell which set of edges corresponds to the border rows by simply observing the graph. By removing the edge $\{(9, 12)\}$, we have disconnect subsystems; thus, the border rows S are associated with $d_{9,12}$ and $d_{12,9}$ ($|S| = 2$). Subsequently, we can identify two cluster matrices $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ corresponding to the sensors contained in each circle in Figure 12, respectively. The $\mathbf{X}^{(1)}$ is a 80×36 matrix, and $\mathbf{X}^{(2)}$ is a 76×36 matrix.

Since $\binom{|R|}{2|S|-2} = \binom{154}{2}$ is less than K, the constant used in Algorithm 4 (recall we set $K = 10^6$), Algorithm 4 tests the rank of $\mathbf{X}_{(-d)}$ for $d = 1$ to $2|S| - 2 (= 2)$ and finds that $\eta(\mathbf{X}) \geq 2|S| - 2$. Then, we can apply Theorem 6 so that $\eta(\mathbf{X})$ should be the smaller one of the calibration redundancies of the two cluster matrices. Testing the ranks of $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ is much faster, and we get $\eta(\mathbf{X}^{(1)}) = 4$ and $\eta(\mathbf{X}^{(2)}) = 4$. By Theorem 6, $\eta(\mathbf{X}) = 4$. The computation of $\eta(\mathbf{X})$ by the decomposition algorithm took less than two hours going through about three millions rank-testings. The number of rank-testing operations is only about 1% of that in the best case scenario for the enumerative algorithm.

To illustrate the robustness of the LTS estimator, we simulate 100 instances of the calibration process using the above sensor network and compare the mean of squared errors (MSE) of the parameter estimation with an ordinary LS estimator. We use two methods to compute the LTS estimator; one is to run the FAST-LTS routine [71] in R

Table V. MSE and computation time of the example in Figure 13

Number of corruptions	LS	FAST-LTS	LTS using sub-model
0	.0681	.0817	.0814
1	.4277	.0792	.0764
2	.5067	.1094	.0804
Time for one iteration	.06 sec.	263.51 sec.	13.13 sec.

on Model (1.1) and the second is to run the same routine on the sub-calibration models in (5.5). We assume that a distance measurement in $\mathbf{y}$ includes a small measurement device error, normally distributed with a zero mean and a standard deviation of .002. In addition, we assume that a distance measurement in $\mathbf{X}$ includes a relatively large measurement error with a zero mean and a standard deviation of .01. We simulate the corrupted distance measurements due to sensor or communication failures by adding a substantial deviation (up to 100%) to some of the measurements in $\mathbf{X}$.

Since $\tau_{\max} = 2$, we simulate the cases with no, one, and two corrupted measurements, respectively. Table V summarizes the MSE's and the computation time of the LTS estimators and the LS estimator. With the presence of data corruptions, the LTS estimator is more robust than the LS estimator, as indicated by its relatively flat MSE value, whereas the MSE values of the LS estimator escalate rapidly. The former's MSE is about one-fifth of the latter's. Regarding computation, an LTS estimation is obviously much more expensive than an LS estimation. But it is worth noting that utilizing the cluster structure in the network can remarkably reduce the computation of an LTS estimator – the LTS estimation using the sub-calibration model consumes about only 5% of the time using the original model (1.1), while the supposed robustness of an LTS estimator is by and large maintained.

The second example concerns a network twice larger (comprising $m = 40$ sensors) than the first example. The communication limit is the same as before. The graph representation is shown in Figure 14, where 159 edges can be observed. There are four micro-calibrated anchor nodes in this network so that $p = 144$ parameters are associated with the remaining 36 ordinary sensors. The calibration matrix $\mathbf{X}$ is thus of 318×144. The size of this matrix makes it almost impossible to use the enumerative rank testing algorithm for computing the redundancy degree. It is equally difficult to use Algorithm 2 or to compute the exact redundancy degree in this case. To illustrate this, consider the following. Take four border rows corresponding to the edges $\{(1, 12)\}$ and $\{(24, 35)\}$. We then partition $\mathbf{X}$ into $k = 3$ blocks and $|S| = 4$ border rows. Algorithm 2 would have to test a total of $\sum_{d=1}^{2|S|-2} \binom{318}{d} \simeq 1.4 \times 10^{12}$ ranks of reduced matrices of $\mathbf{X}$ before testing any cluster matrices.

For a system of this size, it is safer to start with Algorithm 4, which decomposes the original matrix recursively. The first step is to choose either one of the edges $\{(1, 12)\}$ and $\{(24, 35)\}$; let's choose $\{(1, 12)\}$, to partition the graph and the corresponding matrix. We end up with two border rows ($|S| = 2$) and two cluster matrices, $\mathbf{X}^{(1)}$ of 80×36 and $\mathbf{X}^{(2)}$ of 240×108, which correspond to C1 and the collection of C2, C3, and C4 in Figure 14. Since $\binom{|R|}{2|S|-2} = \binom{318}{2} = 50,403 < K$, we test the rank of $\mathbf{X}_{(-d)}$ for $d = 1, 2$ and find that $\eta(\mathbf{X}) \geq 2$. After that, we need to find the minimum of $\nu(\mathbf{X}^{(1)})$ and $\nu(\mathbf{X}^{(2)})$ as stated in the last line of the function Lowerbound$(\mathbf{X}, d)$ in Algorithm 4.

Secondly, for the first cluster matrix $\mathbf{X}^{(1)}$, it is no longer beneficial to decompose further since the sensors within the cluster are densely connected; its redundancy degree is computed by simply testing the ranks of the reduced matrices $\mathbf{X}^{(1)}_{(-d)}$. Given its much smaller size, the computation associated with $\mathbf{X}^{(1)}$ is very much affordable. We find that $\eta(\mathbf{X}^{(1)}) = 4$. The cluster matrix $\mathbf{X}^{(2)}$ can, and should, be decomposed.

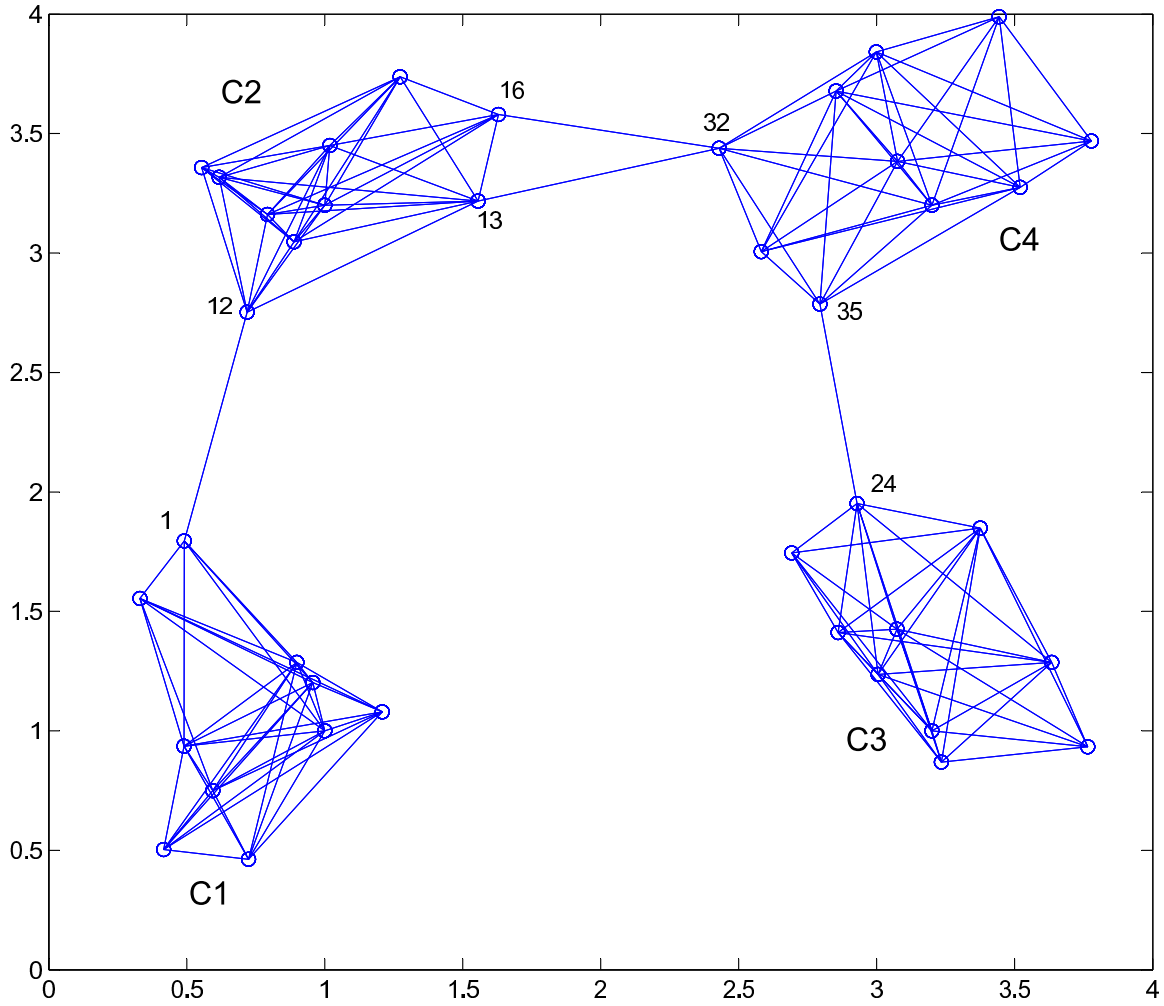


Fig. 14. Graph representation of wireless sensor network (40 sensors)

Now, take the edge $\{24, 35\}$ for the border rows. After decomposing $\mathbf{X}^{(2)}$, we obtain two border rows (i.e., $|S| = 2$) for $\mathbf{X}^{(2)}$ and two second-layer cluster matrices $\mathbf{X}^{(2,1)}$ of 160×72 and $\mathbf{X}^{(2,2)}$ of 82×36. Then, we need to find the minimum of $\nu(\mathbf{X}^{(2,1)})$ and $\nu(\mathbf{X}^{(2,2)})$ like what we did in the first iteration.

Thirdly, it turns out that $\mathbf{X}^{(2,2)}$, which corresponds to C3, needs no decomposition but $\mathbf{X}^{(2,1)}$ does. Testing on $\mathbf{X}^{(2,2)}$, we find $\eta(\mathbf{X}^{(2,2)}) = 4$. Continue carrying out the decomposition on $\mathbf{X}^{(2,1)}$ similar to what was done above. Take the edges $\{13, 32\}$ and $\{16, 32\}$ for the border rows; i.e., there are four border rows (i.e., $|B| = 4$). The third-layer cluster matrices are $\mathbf{X}^{(2,1,1)}$ of size 86×36 and $\mathbf{X}^{(2,1,2)}$ of size 78×36, which correspond to C2 and C4 in Figure 14, respectively. Now, for $\mathbf{X}^{(2,1)}$, $\binom{|R|}{2|S|-2} = \binom{160}{6}$ $\simeq 2 \times 10^{10} \geq K$, so we proceed to find $\nu(\mathbf{X}^{(2,1)})$ using Corollary 13. For that, we obtain $\eta(\mathbf{X}^{(2,1,1)}) = 5$ and $\eta(\mathbf{X}^{(2,1,2)}) = 4$, so $l(\mathbf{X}^{(2,1)}) = \max\{4-4, 0\} + \max\{5-4, 0\} + 1 = 2$. By Corollary 13, $\nu(\mathbf{X}^{(2,1)}) = \min\{\eta(\mathbf{X}^{(2,1,1)}), \eta(\mathbf{X}^{(2,1,1)}), l(\mathbf{X}^{(2,1)})\} = 2$.

Finally, we can get a lower bound by $\eta(\mathbf{X})$ by combining the results given above. We have $\nu(\mathbf{X}^{(2,1)}) = 2$ and $\nu(\mathbf{X}^{(2,2)}) = \eta(\mathbf{X}^{(2,2)}) = 4$. By Corollary 14, a lower bound of $\eta(\mathbf{X}^{(2)})$ is $\nu(\mathbf{X}^{(2)}) = 2$. Using Corollary 14 one more time (with $\nu(\mathbf{X}^{(1)}) = \eta(\mathbf{X}^{(1)}) = 4$), one can get a lower bound of $\eta(\mathbf{X})$ as $\nu(\mathbf{X}) = 2$.

Given this lower bound $\nu(\mathbf{X})$, the trimming parameter in LTS estimation is $h^* = 317$ according to (4.9). Using this h^* to construct an LTS estimator leads to a robust calibration estimate with the fault tolerance capability of $\tau^*(T_{LTS(h^*)}, \mathbf{X}) = 1$. The simulation results of the second example, performed under the same setup of the previous example, are summarized in Table VI. We report a "fail" under the column of FAST-LTS because it failed to compute the LTS estimation after continuously running for two weeks. By comparison, the LTS estimator using the sub-calibration model is much faster and finishes in about half a minute. The MSE of the LTS estimator is considerably lower than that of the LS estimator, when there is one

Table VI. MSE and computation time of the example in Figure 14

Number of corruptions	LS	FAST-LTS	LTS using sub-model
0	.1066	fail	.1215
1	.9569	fail	.1247
2	1.3928	fail	.8348
Time for one iteration	.18 sec.	$>$ two weeks	28.02 sec.

sensor failure or one corrupted measurement; when the number of data corruptions becomes two, MSE of the LTS estimator is not greater than that of the LS estimator. This numerical result is consistent with the theoretical analysis of the fault tolerance capability associated with this LTS estimator (which indicates $\tau^*(T_{LTS(h^*)}, \mathbf{X}) = 1$). The actual redundancy could be higher, but without knowing the exact redundancy degree, it is safer to use the lower bound value that leads to certain robustness.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the conclusions of this dissertation, and clarifies the overall findings derived from the research. Furthermore, this chapter includes possibilities for future research.

A. Conclusions

This dissertation discusses two aspects on the robustness of clustered sensor networks: robust estimation/monitoring procedure using LTS estimation and measuring the fault tolerance capability of a sensor network. These two topics have been studied in engineering and statistics literature, but the overarching study that connects these two areas has not be reported. This dissertation connects the redundancy degree proposed in engineering literature with the robustness measure and the robust estimation procedure that have been intensively studied in statistics literature. In addition to that, the author finds an important connection between the redundancy degree and the cogirth of a vector matroid. The equivalence of the redundancy degree and the cogirth allow us to analyze the computation complexity of the redundancy degree problem, and to develop efficient algorithms under the framework of matroid theory.

1. Finding the redundancy degree

Due to the complexity of the problems of finding the redundancy degree, the author's research is mainly focused on a clustered sensor network. Basically, by utilizing a cluster pattern in a sensor network (i.e., a design matrix $\mathbf{X}$ in a bordered block form), we can find efficiently the redundancy degree from smaller sub-networks of a

given clustered sensor network.

In order to develop efficient algorithms for the redundancy degree, the author applies matroid theory and derives several important theorems for matroid decomposition. Using these theorems, Algorithm 2, 3, and 4 in Chapter III are developed. The author believes that the theorems and algorithms contribute to discrete mathematics, because an efficient algorithm for finding the girth and cogirth of a vector matroid has been needed for several decades [57].

2. Robustness measure

Measuring robustness of a sensor network ensures the monitoring mission of a sensor network in presence of sensor failures. In engineering literature, reliability or redundancy degree has been used for robustness measure. This dissertation proposes the fault tolerance capability defined using the breakdown point in statistics literature for the robustness measure of a sensor network.

3. Robust estimation using LTS estimator

This dissertation provides how to use an LTS estimator for robust estimation results. LTS estimation can attain the maximum fault tolerance capability if the trimming parameter is set in optimal range $h_L \leq h \leq h_U$. This dissertation also includes discussion on the results using $h < h_L$ or $h > h_U$. If one selects $h < h_L$, the resulting LTS estimation loses robustness, and the possibility of breakdown can be high. When $h > h_U$, the resulting LTS estimation has less fault tolerance capability than the optimally tuned LTS estimator; however, the estimation attains a certain level of robustness. In this regard, this dissertation presents LTS estimation using a lower bound of the redundancy degree.

4. Case study

This dissertation presents a multi-station assembly process and wireless sensor networks. Using examples, this dissertation shows that one can gain certain robustness against measurement corruptions and/or violation of model parameters. In fact, an LTS estimator is able to provide robustness in estimation against other types of disturbances such as violation of normality and model uncertainty; please refer to [61] for more details.

B. Suggestions for future work

This section suggests future research directions and discusses issues related to the robustness study presented in this dissertation.

1. Design of a sensor network maximizing fault tolerance capability

The low fault tolerance capability observed in the examples in Chapter V raises a question. How much higher can a fault tolerance capability go if the network configuration is altered or optimized? Maximizing the fault tolerance capability of a sensor network is computationally expensive, and more challenging than evaluating the network's fault tolerance capability. Maximizing the fault tolerance capability of a sensor network by altering network design is where the future research can be directed.

2. Considering different probabilities of sensor failures

This dissertation shows that the maximum fault tolerance capability is half of the redundancy degree. To reach that conclusion we need to assume that all the sensors are equally likely to fail. In other words, no sensor is more reliable than others. To

see this, think about measuring the temperature of a glass of water. Suppose we have two identical thermometers measuring the same glass of water; one says 50 degree and the other says 80 degree. Without further investigation, we cannot tell which thermometer tells the truth. With two thermometers, the redundancy degree is one. Under the assumption they are equally likely to fail, the fault tolerance capability is actually zero. This is consistent with our intuition in the example that we are not sure which thermometer is correct. We need at least another thermometer to break the tie. With three or more thermometers, the redundancy degree is greater than one, and the fault tolerance capability is greater than zero; this means we can still reach the right conclusion even if one sensor failure occurs.

However, the conclusion about the fault tolerance capability does not hold if some sensors are more reliable than others because the fault tolerance capability is purely based on the redundancy degree, which represents the number of sensors. Suppose the reliabilities of sensors are different; then, the reliability information of individual sensors, in addition to the number of sensors, must play a role in determining the final fault tolerance capability. Studying the fault tolerance capability of a sensor network, of which the individual sensor reliabilities are different, is also a possible extension of the research presented in this dissertation.

3. Sensor network consisting of heterogeneous sensors

The implicit assumption made in this dissertation is that we use a set of homogenous sensors, which provide the same type of measurements. However, heterogeneous sensors that provide complementary information can also be used as a safeguard against sensor failures. Heterogeneous sensor networks need an extension of the analysis presented in this dissertation to consider different types of information. Again, robustness analysis for a heterogeneous sensor network is an interesting extension of

the research in this dissertation.

REFERENCES

[1] R. S. Mah, G. M. Stanley, and D. M. Downing, "Reconciliation and rectification of process flow and inventory data," *Industrial & Engineering Chemistry Process Design and Development*, vol. 15, pp. 175–183, 1976.

[2] H. C. Turbatte, D. Maquin, B. Cordier, and C. T. Huynh, "Analytical redundancy and reliability of measurement system," in *Proceedings of IFAC Symposium on Safety of Technical Processes*, Baden-Baden, Germany, 1991, pp. 49–54.

[3] S. J. Hu and S. M. Wu, "Identifying root cause of variation in automobile body assembly using principal component analysis," *Transactions of NAMRI*, vol. 20, pp. 311–316, 1992.

[4] J. S. Carlson, L. Lindkvist, and R. Soderberg, "Multi-fixture assembly system diagnosis based on part and subassembly measurement data," in *Proceedings of 2000 ASME Design Engineering Technical Conference*, Baltimore, MD, 2000, pp. 10–13.

[5] Y. Ding, D. Ceglarek, and J. Shi, "Fault diagnosis of multi-station manufacturing processes using state space approach," *ASME Journal of Manufacturing Science and Engineering*, vol. 124, pp. 313–322, 2002.

[6] L. Mili, M. G. Cheniae, and P. J. Rousseeuw, "Robust state estimation of electric power systems," *IEEE Transactions on Circuits and Systems*, vol. 41, pp. 349–358, 1994.

[7] R. Dorr, F. Kratz, J. Ragot, F. Loisy, and Cermain J., "Detection, isolation, and identification of sensor faults in nuclear power plants," *IEEE Transactions on Control Systems Technology*, vol. 5, pp. 42–60, 1997.

[8] R. N. Clark, "Instrument fault detection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-14, pp. 454–465, 1978.

[9] G. M. Stanley and R. S. H. Mah, "Observability and redundancy in process data estimation," *Chemical Engineering Science*, vol. 36, pp. 259–272, 1981.

[10] M. Luong, D. Maquin, C. T. Huynh, and J. Ragot, "Observability, redundancy, reliability and integrated design of measurement systems," in *Proceedings of 2nd IFAC Symposium on Intelligent Components and Instruments for Control Applications, SICICA 94*, Budapest, Hungary, 1994.

[11] M. Staroswiecki, G. Hoblos, and A. Aïtouche, "Sensor network design for fault tolerant estimation," *International Journal of Adaptive Control and Signal Processing*, vol. 18, pp. 55–72, 2004.

[12] A. Kretsovalis and R. S. H. Mah, "Observability and redundancy classification in generalized process networks - i. theorems," *Computers and Chemical Engineering*, vol. 12, pp. 671–687, 1988.

[13] A. Kretsovalis and R. S. H. Mah, "Observability and redundancy classification in generalized process networks - ii. algorithms," *Computers and Chemical Engineering*, vol. 12, pp. 689–703, 1988.

[14] Y. Ali and S. Narasimhan, "Sensor network design for maximizing reliability of linear processes," *AIChe Journal*, vol. 39, pp. 820–828, 1993.

[15] Y. Ali and S. Narasimhan, "Redundant sensor netowrk design for linear processes," *AIChe Journal*, vol. 41, pp. 2237–2249, 1995.

[16] J. Levine and R. Marino, "On fault-tolerant observers," *IEEE Transactions on Automatic Control*, vol. 35, pp. 623–627, 1990.

[17] J. G. Webster, *Measurement, Instrumentation and Sensors Handbook*, CRC Press, Boca Raton, FL, 1998.

[18] B. R. Updahyaya, "Sensor failure detection and estimation," *Journal of Nuclear Safety*, vol. 26, pp. 23–32, 1985.

[19] M. P. Henry and D. W. Clark, "The self-validating sensor: Rationale definitions and examples," *Control Engineering Practice*, vol. 1, pp. 585–610, 1993.

[20] M. P. Henry, "Plant asset management via intelligent sensors - digital, distributed and for free," *Computing and Control Engineering Journal*, vol. 8, pp. 284–304, 2000.

[21] G. Betta, "Instrument fault detection and isolation: State of the art and new research trends," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, pp. 100–107, 2000.

[22] I. F. Akyildiz, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[23] H. M. F. Aboelfotoh, S. S. Iyengar, and Krishnendu Chakrabarty, "Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures," *IEEE Transactions on Reliability*, vol. 54, pp. 145–155, 2005.

[24] D. N. Jayasimha, "Fault tolerance in multisensor network," *IEEE Transactions on Reliability*, vol. 45, pp. 308–320, 1996.

[25] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Transactions on Computer Systems*, vol. 8, pp. 284–304, 1990.

[26] D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap, "Information integration and synchronization in distributed sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 1032–1043, 1991.

[27] S. S. Iyengar and L. Prasad, "A general computational framework for distributed sensing and fault-tolerant sensor integration," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 1032–1043, 1995.

[28] P. J. Huber, "Robust regression: Asymptotics, conjectures and Monte Carlo," *Annals of Statistics*, vol. 1, pp. 799–821, 1973.

[29] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the American Statistical Association*, vol. 69, pp. 383–393, 1974.

[30] W. S. Krasker, "Estimation in linear regression models with disparate data points," *Econometrika*, vol. 48, pp. 1333–1346, 1980.

[31] W. S. Krasker and R. E. Welsch, "Efficient bounded influence regression estimation," *Journal of the American Statistical Association*, vol. 77, pp. 595–604, 1982.

[32] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, pp. 871–880, 1984.

[33] G. W. Basset, Jr, "Equivariant, monotonic, 50% breakdown estimators," *The American Statistician*, vol. 45, pp. 135–137, 1991.

[34] M. Tableman, "The asymptotics of the least trimmed absolute deviations (LTAD) estimator," *Statistical Probability Letters*, vol. 17, pp. 387–398, 1994.

[35] J. Jureckova and S. Protnoy, "Asymptotics for one-step m-estimators with application to combining efficiency and high breakdown point," *Communications in Statistics - Theory and Methods*, vol. 16, pp. 2187–2199, 1987.

[36] V. J. Yohai, "High breakdown point and high efficiency robust estimates for regression," *Annals of Statistics*, vol. 15, pp. 642–656, 1987.

[37] V. J. Yohai and R. H. Zarmar, "High breakdown point estimates of regression by means of the minimization of an efficient scale," *Journal of the American Statistical Association*, vol. 83, pp. 406–414, 1988.

[38] C. W. Coakley and P. Hettamansperger, "A bounded influence, high breakdown, efficient regression estimator," *Journal of the American Statistical Association*, vol. 88, pp. 872–880, 1993.

[39] D. L. Donoho and P. J. Huber, "The notion of breakdown point," in *A Festschrift for Erich L. Lehman*, P. Bickel, K. Doksum, and J. L. Hodges Jr., Eds., pp. 157–184. Wadsworth, Belmont, CA, 1983.

[40] P. L. Davies, "Aspects of robust linear regression," *Annals of Statistics*, vol. 21, pp. 1843–1899, 1993.

[41] L. Mili and C. W. Coakley, "Robust estimation in structured linear regression," *Annals of Statistics*, vol. 24, pp. 2593–2607, 1996.

[42] S. Narasimhan and R. S. H. Mah, "Generalized likelihood ratio method for gross error identification," *AIChE Journal*, vol. 33, pp. 1514–1521, 1987.

[43] C. M. Crowe, "Recursive identification of gross errors in linear data reconciliation," *AIChE Journal*, vol. 34, pp. 541–550, 1988.

[44] M. Bagajewicz and Q. Jiang, "Gross error modelling and detection in plant linear dynamic reconciliation," *Computer and Chemical Engineering*, vol. 22, pp. 1789–1809, 1998.

[45] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, *Exploring Data Tables, Trends, and Shapes*, John Wiley & Sons, Hoboken, NJ, 1985.

[46] M. C. Ferris and J. D. Horn, "Partitioning mathematical programs for parallel solution," *Mathematical Programming*, vol. 80, pp. 35–61, 1998.

[47] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, "Permuting sparse rectangular matrices into block-diagonal form," *SIAM Journal on Science Computing*, vol. 25, pp. 1860–1879, 2004.

[48] G. Karypis, V. Kumar, R. Aggarwal, and S. Shekhar, *A Hypergraph Partitioning Package Version 1.0.1*, Department of Computer Science and Engineering/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 1998.

[49] Ü. V. Çatalyürek and C. Aykanat, *PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0*, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 1999.

[50] S. Even, *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.

[51] K. Menger, "Zur allgemeinen kurventheorie," *Fundamenta Mathematicae*, vol. 10, pp. 96–115, 1927.

[52] J. G. Oxley, *Matroid Theory*, Oxford University Press, New York, NY, 1992.

[53] J. Lee and J. Ryan, "Matroid applications and algorithms," *ORSA Journal on Computing*, vol. 44, pp. 70–98, 1992.

[54] O. R. Oellermann, "Connectivity and edge-connectivity in graphs: A survey," *Congressus Numerantium*, vol. 116, pp. 231–252, 1996.

[55] D. W. Matula, "Determining edge connectivity in $O(nm)$," *28th Annual Symposium on Foundation of Computer Sciences*, pp. 249–251, 1987.

[56] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM Journal on Discrete Mathematics*, vol. 5, pp. 54–66, 1992.

[57] D. J. A. Welsh, *Combinatorial Problems in Matroid Theory*, Academic, London, U.K., 1971.

[58] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Transactions on Information Theory*, vol. 43, pp. 1757–1766, 1997.

[59] L. Khachiyan, E. Boros, K. Elbassioni, V. Gurvich, and K. Makino, "On the complexity of some enumeration problems for matroids," *SIAM Journal on Discrete Mathematics*, vol. 19, no. 4, pp. 966–984, 2006.

[60] P. D. Seymour, "A note on hyperplane generation," *Journal of Combinatorial Theory, Series B*, vol. 61, pp. 88–91, 1994.

[61] R. R. Wilcox, *Introduction to Robust Estimation and Hypothesis Testing*, Academic Press, San Diego, CA, 2005.

[62] R. Maronna, D. Martin, and V. Yohai, *Robust Statistics - Theory and Methods*, John Wiley & Sons, Hoboken, NJ, 2006.

[63] Y. Chen, "Application of matroid theory for diagnosability study of coordinate sensing systems in discrete-part manufacturing processes," *Technometrics*, vol. 48, pp. 386–398, 2006.

[64] J. Jin and J. Shi, "State space modelling of sheet metal assembly for dimensional control," *ASME Journal of Manufacturing Science and Engineering*, vol. 121, pp. 756–762, 1999.

[65] P. Enge and P. Misra, "Special issue on global positioning system," *Proceedings of the IEEE*, vol. 87, pp. 3–172, 1999.

[66] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," *IEEE Personal Communications*, vol. 8, pp. 52–59, 2001.

[67] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of Mobile Computing and Networking*, 2000.

[68] J. Hightower, "The location stack," Ph.D. dissertation, Department of Computer Science & Engineering, University of Washington, Seattle, WA, 2004.

[69] K. Whitehouse and D. Culler, "Macro-calibration in sensor/actuator networks," *Mobile Networks and Applications*, vol. 8, pp. 463–472, 2003.

[70] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, 2002, pp. 59–67.

[71] P. J. Rousseeuw and K. Van Driessen, "Computing LTS regression for large data sets," *Data Mining and Knowledge Discovery*, vol. 12, pp. 29–45, 2006.

[72] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a netowrk with power estimation," *Soviet Mathematical Doklady*, vol. 11, pp. 1277–1280, 1970.

[73] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, John Wiley & Sons, Hoboken, NJ, 1998.

APPENDIX A

MENGER'S THEOREM AND EVEN'S ALGORITHM

Menger's Theorem

Menger's theorem in this dissertation follows [50]. Let $G(V, E)$ be a graph with a vertex set V and a edge set E, and there is no loop or parallel edge in G. Suppose $\{a, b\} \in V$; $(a, b)-$separator S is defined as a set of vertices such that every path connecting a and b passes through a vertex in S. Denote by $N(a, b)$ the least cardinality of an $(a, b)-$separator, and $p(a, b)$ the maximum number of pairwise vertex-disjoint paths between a and b.

Theorem 17 (Menger's Theorem) *If there is no edge connecting a and b, then*

$$N(a, b) = p(a, b)$$

This theorem can be easily proved using min-cut max-flow theorem, so the proof is omitted (for those who need the proof, please refer to page 121 in [50]).

Even's Algorithm

Based on Menger's Theorem, Even developed an algorithm to find a minimum separating set [50]. A summary of the algorithm is presented as follows.

Again, suppose we have a graph $G(V, E)$. First, construct a digraph $\bar{G}(\bar{V}, \bar{E})$ as follows. For every vertex $v \in V$, put two vertices v' and v'' in $\bar{V}$, and put directed edge (v', v''). For every edge $e = \{u, v\}$ in G, put two directed edges $e' = (u'', v')$ and $e'' = (u', v'')$ in $\bar{E}$. Then, assign unit capacities for all the internal edges (for example, (v', v'')) and assign infinite capacities for all e' and e'' (e' and e'' are called

external edges). Now, we have a network; the max flow of the network is in fact the size of the minimum separating set of G, and the minimum cut of the network is equivalent to the separating set of G. In [50], Dinic' Algorithm [72] is used for finding the minimum cut, but users can select any algorithms for finding the minimum cut. For more about the minimum cut problem, please refer to page 37–61 in [73].

VITA

Name:	Jung Jin Cho
Address:	Industrial and Systems Engineering, 3131 TAMU, College Station, TX 77840-3131
Email Address:	drjjcho@gmail.com
Education:	B.S., Industrial Engineering, Seoul National University, 1997 M.S., Industrial Engineering, Seoul National University, 2001